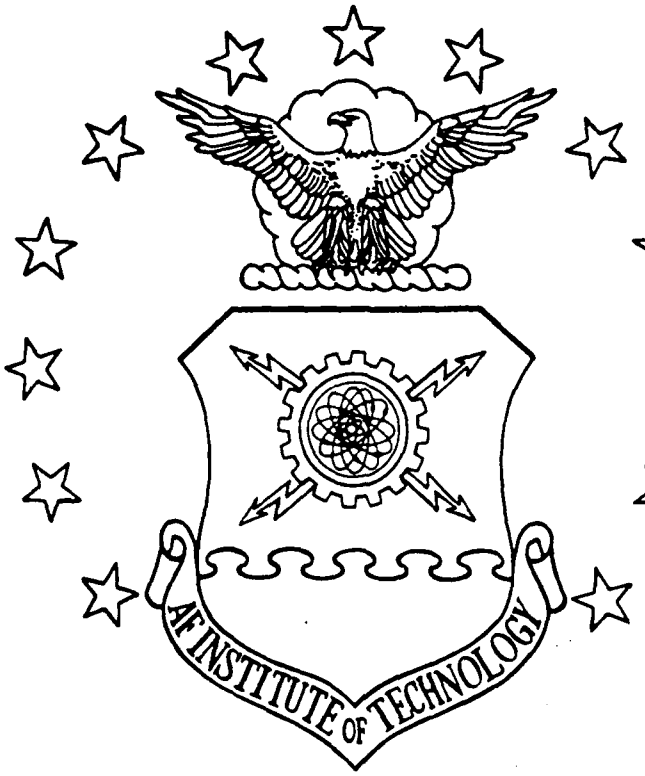

MODELLING THE SCHEDULED PREVENTIVE MAINTENANCE AS A LINEAR SYSTEM

THESIS

M. ERDOGAN GUNES
1st.Lt., TURKISH AIR FORCE

AFIT/GOR/AA/88D-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GOR/AA/88D-01

MODELLING THE SCHEDULED PREVENTIVE MAINTENANCE AS A LINEAR SYSTEM

THESIS

M. ERDOGAN GUNES
1st.Lt., TURKISH AIR FORCE

AFIT/GOR/AA/88D-01

Approved for public release; distribution unlimited

AFIT/GOR/AA/88D-01

MODELLLING THE SCHEDULED PREVENTIVE MAINTENANCE AS A LINEAR SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

M. Erdogan Gunes

1st. Lt., Turkish Air Force

December 1988

Preface

During the lifecycle of a system, a machine, a plant or any facility; effectiveness of preventive maintenance, performed on that system is the most important fact that bounds the reliability,operating cost and useful life of the system. And the effect of the preventive maintenance on the system should be studied during the design phase. Therefore, models to predict the future reliability of a system, under different scheduled preventive maintenance policies, are needed by the design engineers during the early design and planning processes. The purpose of this research is to develop a model of scheduled preventive maintenance that will allow the design engineer and logistic planners to predict future, long term system reliability based on scheduled preventive maintenance with different maintenance periods and capabilities. The model uses Markov processes but formulates the equations as a linear, state variable control system.

This technique will simplify the Markov models of large and complex systems. And will lead to a computer model which runs fast, has low operating cost and consequently provides higher sensitivity analyses.

The only research project which uses this technique to evaluate the system reliability under preventive maintenance was done by Captain Gregory R. Miller in March 1988.

His study on a single component system showed that using a linear state, control system in formulation would simplify the rather large, complex models which usually accompany Markov models. I would like to acknowledge the help of his study on expanding the technique to multicomponent systems. And I would especially thank to my thesis advisor, Major David G. Robinson. His advice and assistance has been invaluable.

M. Erdogan Gunes

Table of Contents

List of Figures

List of Tables

Abstract

Reliability, operating cost and useful life of a system is depend on the effectiveness of the preventive maintenance performed over the life cycle of the system. Another very important factor on long term reliability and overall supportability of systems is the contact and information flow between design engineers and logistic planners. The purpose of this research is to develop a model of scheduled preventive maintenance that will allow the design engineer and logistics planners to predict future, long term system reliability based on scheduled preventive maintenance with different maintenance periods and capabilities. The model uses Markov processes but formulates the equations as a linear, state variable control system. This technique will simplify the Markov models of large and complex systems. And will lead to a computer model which runs fast, has low operating cost and consequently provides higher sensitivity analyses.

MODELLING THE SCHEDULED PREVENTIVE MAINTENANCE AS A LINEAR SYSTEM

I Introduction

Reliability, operating cost and useful life of a system depend on the effectiveness of preventive maintenance, performed over the life cycle of the system. During the design process of a system, a main concern of the design engineer is the decreasing reliability and subsequent scheduling of preventive maintenance. While keeping the production of the system on-time and on-budget the reliability of the system should meet the customer's requirements. This is only possible by ensuring the reliability requirements during the design process. After the system is in the field its reliability will start decreasing, no matter how good it is designed. The solution to insure that the system's reliability remains above a certain minimum level during its useful life is periodic preventive maintenance.

> Preventive maintenance(PM) is any pre-scheduled task or activity that is performed on an operational system or facility with one of three objectives in mind:
> 1. to prevent equipment deterioration and failure,
> 2. to detect incipient failure, and
> 3. to discover hidden failures in off-line systems before an operating demand is made. Effective PM involves all three objectives. All PM programs address the first objective(10:120).

With the knowledge of desired useful life and projected maintenance effectiveness, the design engineer could find approximate preventive maintenance intervals required to maintain a minimum reliability standard.

There are six methods to predict systems reliability performance: Markov, piecepart count, network, Monte Carlo, fault tree and decision analysis(3:125). Of these methods, Markov analysis is the only low cost method for analyzing complex systems and evaluating maintenance strategies. Piece part counts are useful for no maintenance, nonredundant hardware systems. Trees and network analysis are not useful for anlyzing maintenance strategies. And Monte Carlo simulations are expensive and very inefficient for sensitivity analysis. This research will use Markov analysis to model the preventive maintenance scheduling.

The Markov modeling provides a lot of advantages but it is not free of weaknesses. Markov analysis of a system consists of a set of components and several possible states for each component. The problem is that the total number of states in the model grows exponentially as the number of components in the system increase, e.g. in a system with N components if the characteristics of each component is modeled by M states then the model would have total of M^N states. This state growth problem makes the method inefficient for low level analysis of very complex systems.

Statement of The Problem

Models to predict the future reliability of a system, under different scheduled preventive maintenance policies are needed by design engineers during the early design and planning process. Such a model easy analysis of different maintenance strategies and the operating cost of the model should be low, this extensive sensitivity analyses become possible. The only low-cost method which can be used on the analysis of maintenance strategies is the Markov processes. But the state growth problem limits the maximum number of components in the system.

Research Question

To simplify the "state growth" problem of Markov processes; is it possible to use a state-space linear system formulation with impulse inputs to model the preventive maintenance of a system by Markov processes?

Subsidiary Questions

1. What is the connection between elements of transition matrix of Markov processes and the elements of linear system formulas?

2. Does the approach of linear state-space formulation reduce the number of states in Markov model of a system?

3. Does the linear state-space formulation reduce the number of computations in Markov model?

Research Objectives

This research will investigate a state-space linear system formulation for modeling the preventive maintenance of a system using Markov processes.

The research will first focus on one component of the system which has three states representing the operating conditions of component considering preventive maintenance. Then the model will be generalized to multicomponent systems where each component of the system has three states.

Design of an interactive computer model which can be used by design engineers on "what-if" type studies will be the main goal of the research. According to the formulation steps, first an experimental program for a single component will be designed. And then the final program for multicomponent systems will be developed. The final program should:

1. be able to read system parameters either from a data file or directly from keyboard.

2. produce a graphical output that shows the system reliability during the given simulation time.

3. allow the user to run the same system with different parameters without leaving the program (Ability to do quick sensitivity analysis).

4. produce an output file which the system reliability is recorded in time periods given by user.

Overview

Since this modeling technique is very new area a literature review in general sense is not possible. But the methods to predict the system reliability are addressed in the following chapter, Chapter II. The advantages and disadvantages of different methods compared and the reasons to choose the Markov model for preventive maintenance are listed.

Supported by the theory, concepts and assumptions of Markov processes, Chapter III presents the methodology used in formulation of linear state-space modeling of single component. The implementation of this model is demonstrated by some example problems.

Chapter IV attempts to generalize the model to multicomponent systems, and explains the system states along with definition of the method to build the system matrixes. This chapter also introduces some example problems to demonstrate the program.

The summary and conclusions of this research is represented in chapter V. Implication of the model and further research ares are also pointed.

The computer source codes are included as appendices to this report.

II. Background:

Major Methods For System Reliability Prediction And Markov Models

Reliability prediction of a new system during the planning and design phases has become very important in both industry and government. To obtain maximum reliability in the later design and production stages, effective prediction methods are needed by design engineers.

The important characteristics for a model are closed-form solution, low operating cost and sensitivity analyses. Powerful models should provide system output from component input and permit closed-form mathematical solutions which has lower operating costs and higher ability of sensitivity analysis.

Most of the methods for predicting system reliability fall into six classifications(3:125). These are Markov, piecepart count, network, Monte Carlo, fault tree and Decision analysis. Markov, piecepart count, network and fault tree methods are based on closed-form solutions of mathematical formulas and operate at relatively low cost. Monte Carlo methods are more expensive, but have the capability to more closely model specific system details(3:125). Capabilities of four model are given by ref.(5:431) and repeated in Table1.

Capability	Markov	Piecepart	Trees	M. Carlo
Low operating cost	*	*	*	
Sensitivity	*	*	*	
Coverage	*		*	*
Time dependent I/O	*			*
Generality of system structure	*		*	*
Detailed system factors modeling	*			*
Maintainability related O&S cost prediction	*			*

Table 1. Reliability prediction methods overview(5:431)

Piecepart Count

Piecepart count method is used on nonredundant hardware systems and this involves estimating component failure rates and counting the number of components(5:427). Then the system reliability is a weighted sum of part failure rates. Some properties of complex or fault-tolerant systems such as imperfect failure coverage between redundant components, preventive maintenance and non series parallel system structure makes the piecepart count method inappropriate(5:427).

Monte Carlo

The same reference(5:427) explains the advantage and disadvantages of Monte Carlo models as follows:

> Monte Carlo methods most closely model specific system details. On the other hand, the extensive sensitivity analysis necessary for early stage system design and logistics policy trade-offs is difficult using Monte Carlo methods. Often the input data necessary to run simulations is not available during early design. This method is useful for modeling requirements in detail, but does not allow low cost testive predictions involving design uncertainty.

Fault Tree Analysis

One of the low-cost methods modeling complex system structures is network or fault tree analysis. In a fault tree analysis, a system is represented by a fault tree which corresponds to the possible event sets leading to system failure. The name "fault tree" may cause some misunderstandings, the events in the fault tree are not limited to "faults" but denote various happenings, outcomes or conditions which were identified as relevant factors for the occurrence of the top event(11:310). This can be a system malfunction, failure to achieve a certain goal, or other undesirable occurrences.

A typical tree may consist of several hundred events, and it may be very time consuming to evaluate the entire fault tree. The first step in a fault tree analysis is to reduce the fault tree into minimal cutsets which are combinations of basic events that lead to system failure. Therefore the analysis is initiated by two data sets(8:359):

1. A list of cut sets for each unique system failure, and

2. The basic events and associated failure probabilities.

The system failure probability may be evaluated by first calculating the probability of each cut set by multiplying the basic event probabilities in that cut set, and then estimating the system failure probability by summing the probabilities of all the cut sets.

> The use of fault trees offers major benefits for the design of complex systems consisting of several cross-linked subsystems with contradictory design requirements, particularly when the design requires involvement of various specialists. Typical examples of such systems are nuclear reactors and microwave communication networks. The fault tree has only limited application for simple systems and for systems in which the complexity is caused by multiple applications of identical building blocks.(11:310).

Although the fault tree or network methods are very good for complex system structures, they are not capable adressing maintenance strategies. When predicting the reliability of a repairable system, modeling the maintenance activity is very important. The only non Monte Carlo modeling technique which permits maintenance strategy analyses is the Markov method(5:427).

Markov Models

Markov modeling is a flexible, graphically-assisted technique useful in quantitative "design for reliability" decision making, assessing conformance to reliability objectives, and optimization of maintenance strategies for complex systems(6:290).

A Markov model consists of a number of system states and transitions between them. Each state is defined by a state vector, where each element of the vector takes on an integer value within a defined range. An element can represent any meaningful characteristic, such as the number of good components of one type in the system, or the number of faulty components of another type in use(4:17). Therefore in a Markov model, a system is represented by a collection of states and a probability model which defines the probability of whether being in any particular state or transitioning from one state to another.

The three basic assumptions in Markov models are:

1. The transitional probability from one state to another is a function of time interval's length, not of the time of transition(a homogeneous, stationary Markov chain).

2. The transitional probability is independent of past transitions(memoryless property).

3. The probability of more than one transition in any interval is very small(a Poisson process).

The main drawback in Markov models is that the number of states grows exponentially according to number of components in the system. For example in a system in which every component has two states -either operational or failed- the number of states is 2^N where N is the number of components. And it is 3^N if a single component has three states, etc. Thus the ability to represent interdependent maintenance strategies in large systems is very limited(3:125).

But Markov models offer some unique advantages to reliability engineers. These benefits can be classified as in the following list(3:125-126):

1. Stand-alone benefits;
   - Models maintenance and support policy with reliability.
   - Models module dependencies through maintenance or system structure.
   - Ideally suited for "n-of-m" fault-tolerant systems with identical components.
   - Use as system testability model.
2. Enhancement of traditional system reliability evaluation methods (fault trees, networks);
   - Assists in early stage selection of an optimal main tenance strategy when used in conjunction with simulation.
   - Permits cross-including validation of simulation results for simpler maintenance policies.
3. Enhancement of traditional maintainability and logistics support evaluation methods (simulations) to determine optimal maintenance procedures, integrated logistics support (ILS), and warranty/incentive contracts;
   - Markov combines with fault trees or networks in a hierarchial fashion.
   - Markov provides an "in depth" look at key portions of system maintenance dependency.

- Output from component-level failure rate or design for testability models feeds into a Markov model as LRU/WRA level input.

In the next chapter preventive maintenance of a single component will be modeled by the Markov method, accompanied with associatedlinear system, state space formulation.

III. Model For Single Component

Until this point, all major methods and models that are used in system reliability prediction was listed. And the reasons for choosing the Markov technique to model a system's reliability under maintenance were addressed. In this chapter first the Markov states and transitions of a single component will be defined and the solution for a single component's reliability without preventive maintenance will be presented. Then after giving the basic concept of linear system formulation the effect of preventive maintenance will be added to the model. And the linear system formulation technique will be used to simplify the Markov formulation. At the end of the chapter the implementation of computer model will be presented with some example problems.

Markov Processes

The different conditions of a component during it's operational life could be modeled as different states. In our model the conditions of a component will be presented by three states:

1) Perfect, 2) Degraded, 3) Failed. When the component is in state 1 which is called perfect, it is fully operational or as good as new.

State 2 represents a condition of the component which is neither fully operational nor failed.

And the state 3 represents the failed component which can not operate without a maintenance action.

This Markov process models the component's operation by transitions from one state to another. The probabilities of transitioning from one state to another are provided by a time history of the probabilities that the component operates in each of the three states. The sum of the probabilities of being in state 1 and state 2 at any time gives the component's reliability for that time. And the transition matrix built by transition rates yields a solution which gives the component's reliability as a function of time. Three assumptions regulate the transitions. All assumptions, used in this model will be discussed in more detail later, but these assumptions will be given in the following paragraph to produce a more understandable text.

The three assumptions taken from Markov modeling of component's operational life are:

1. Transition probabilities are a function of time interval at which transition occurs, not of the time of transition. Therefore it is a homogeneous, stationary Markov chain(1:112-141).

2. The transitional probabilities are not depend on the number of past transitions(1:112-141).

3. The probability of more than one transition in any interval is very small[9].

The state transition diagram of the component without preventive maintenance is given in Figure 1. Where L1, L2, L3 are the transition probabilities as they are assigned in Table 2.

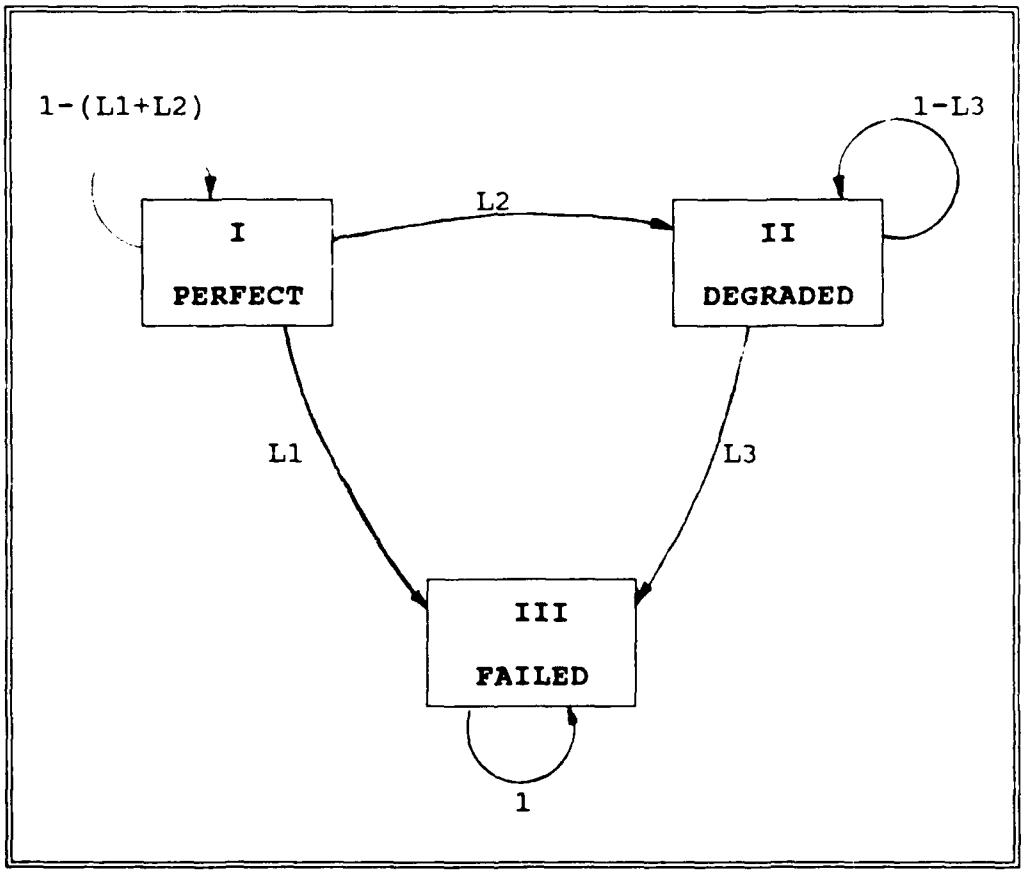

Figure 1. Component's state transition diagram without preventive maintenance.

VALUE	TRANSITION
L1	From perfect to failed
L2	From perfect to degraded
L3	From degraded to failed

Table 2. Transition Probabilities.

Since the only maintenance we are concerned with is the preventive maintenance, once the component fails it remains inoperative. Therefore the State 3 is an absorbing state.

We can now build the component transition matrix using the transition probabilities. And we can find the we find the equation of operating states probabilities as a function of time by using Chapman-Kolmogorov equation(2:272-295). The transition matrix of the component without preventive maintenance is as follows:

$$\underline{T} = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = \begin{bmatrix} 1-(L1+L2) & L2 & L1 \\ 0 & 1-L3 & L3 \\ 0 & 0 & 1 \end{bmatrix} \qquad (1)$$

where, P_{ij} is the transition probability from state (i) to state (j). And $\underline{T}$ will represent the transition matrix of single component without preventive maintenance. This matrix leads to a system of linear homogeneous differential equations with constant coefficients. Solutions can be found either through matrix methods, or through the simultaneous solution of the resulting differential equations.

These differential equations will be derived in the following calculations where $P_{ij}(t)$ represents the probability being in state (i) at time (t):

$$\begin{aligned} P_1(t+dt) &= [1-(L1+L2)dt]P_1(t) + 0(dt) \\ P_2(t+dt) &= [(L2)dt]P_1(t) + [1-(L3)dt]P_2(t) + 0(dt) \\ P_3(t+dt) &= [(L1)dt]P_1(t) + [(L3)dt]P_2(t) + P_3(t)dt \end{aligned} \quad (2)$$

or,

$$\frac{P_1(t+dt) - P_1(t)}{dt} = -(L1+L2)P_1(t)$$

$$\frac{P_2(t+dt) - P_2(t)}{dt} = (L2)P_1(t) - (L3)P_2(t)$$

$$\frac{P_3(t+dt) - P_3(t)}{dt} = (L1)P_1(t) + (L3)P_2(t) \quad (3)$$

or,

$$\dot{P}_1 = \frac{dP_1}{dt} = -(L1+L2)P_1(t)$$

$$\dot{P}_2 = \frac{dP_2}{dt} = (L2)P_1(t) - (L3)P_2(t)$$

$$\dot{P}_3 = \frac{dP_3}{dt} = (L1)P_1(t) + (L3)P_2(t) \quad (4)$$

The solutions to the three equations in (4) can lead to closed-form solution of each state.

The sum of the probabilities of being in State 1(P_1), and State 2(P_2) at any time gives the component's reliability at that time. After adding the effect of preventive maintenance to the model and increasing the number of components in the system, it is not always possible to find a closed-form solution for system reliability by using this differential equations. It is not efficient to seek a closed-form solution for every alternative system either. Since, the purpose of this study is to find a model which works for any system, we will leave this equations in the following form:

$$\begin{bmatrix} \dot{P}_1 \\ \dot{P}_2 \\ \dot{P}_3 \end{bmatrix} = \begin{bmatrix} -(L1+L2) & 0 & 0 \\ L2 & -L3 & 0 \\ L1 & L3 & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \end{bmatrix} \qquad (5)$$

or,

$$\dot{\underline{P}}(t) = (\underline{T} - I)^{T}\underline{P}(t) \qquad (6)$$

where;

$\dot{\underline{P}}(t)$: matrix of state probability rate changes

$\underline{P}(t)$: matrix of state probabilities at time (t)

$\underline{T}$: transition matrix of the component without preventive maintenance

I : Identity matrix in proper dimension with $\underline{T}$.

And the component's reliability at time (t) is:

$$R(t) = P_1(t) + P_2(t) \qquad (7)$$

or,

$$R(t) = [\ 1 \quad 1 \quad 0\ ]\ \underline{P}(t) \tag{8}$$

Now before introducing the effect of the preventive maintenance into the model, we need to look at linear system formulation technique. Because in case of preventive maintenance the right hand side of equation (6) needs more than one term, and the theory behind the linear system formulation will help to determine the new terms.

Linear System State-Space Equations and Solutions

A state space formulation for a linear, time invariant, dynamic system is given by:

$$\dot{X} = AX + BU \tag{9}$$
$$Y = CX + DU \tag{10}$$

For a general n^{th}-order, 1-input, m-output state space model A, B, C, D are matrices with dimensions n x n, n x 1, m x n, and m x 1 respectively(10:246). We will not use the term DU in (10). For the other terms in the formulation: X is the matrix of system states, U is the matrix of inputs and Y is the output matrix.

Solutions to (8) and (9) for continuous and discrete time are given by Reid(7:246-294). The general solution for continuous time and with D = 0 is:

$$X(t) = e^{At}X(0) + e^{At} \int_0^t e^{-A\tau} B\, U(\tau)\, d\tau \qquad (11)$$

and

$$Y(t) = Ce^{At}X(0) + Ce^{At} \int_0^t e^{-A\tau} B\, U(\tau)\, d\tau \qquad (12)$$

where, X(0) is initial state at time t = 0.

As it is seen in (11) and (12) the solution of linear state-space models for continuous time is very complicated, and does not promise too much advantage for this study. But,

> The real power of state-space methods comes with digital analysis and digital simulation(7:245-246).

To see the real power of state-space analysis with the support of digital computers, let us look at the discrete time solution of (9) and (10) given by Reid(7:273):

$$X_T(k) = A_T\, X_T(k-1) + B_T\, U_T(k-1) \qquad (13)$$

$$Y_T(k) = C_T\, X_T(k) + D_T\, U_T(k-1) \qquad (14)$$

where;

$$U_T(k) = U(t) \quad , \quad t \in (kT,\ kT+T), \qquad (15)$$

$$A_T \equiv e^{AT} \ , \qquad (16)$$

$$B_T \equiv \left(\int_0^T e^{A\tau}\, d\tau \right) B \,, \tag{17}$$

and $$C_T = C \,, \; D_T = D \tag{18}$$

> Notice that. . . only the current input and the current state are used to propagate to the next state. All of the necessary memory about the past state values is contained in the present state, and so all past state values and past input values can be discarded. This is fundamental to the state-space approach to digital simulation of the linear invariant system (7:274).

By this discrete time step analysis, equations can be evaluated by computer very efficiently. Because the iterative solution set requires a minimum amount of storage and by choosing proper time step values(T), system output can be calculated for any given time very fast. The only drawback is, the system output value is available only at this discrete time steps.

> But one can never do better than this with the inherently discrete instants of time,kT. But one wants a tighter spacing, one simply makes T shorter(7:271).

Accuracy of the solution is dependent on the calculation of A_T and B_T. The fastest and easiest way of calculation of A_T and B_T is by the power series expansion method(7:274-275):

$$A_T = I + AT + \frac{A^2 T^2}{2!} + \frac{A^3 T^3}{3!} + \cdots \tag{19}$$

$$B_T = \left(IT + \frac{A T^2}{2!} + \frac{A^2 T^3}{3!} + \cdots \right) B \tag{20}$$

or by using a single power series:

$$\boxed{E = IT + \frac{A T^2}{2!} + \frac{A^2 T^3}{3!} + \cdots} \tag{21}$$

Then,

$$\boxed{\begin{aligned} A_T &= I + AE \\ B_T &= EB \end{aligned}} \tag{22}$$

The calculation of A_T and B_T may lead to some discussion about convergence of power series and accuracy of model. There are some methods, however to increase the accuracy of this power series, and the decision belongs to the model user to decide how much accuracy is needed or satisfactory for his purpose.

Now we are ready to look at the effect of preventive maintenance on the component. In the following section preventive maintenance of the component will be introduced into the model and the previous solution given by equations (6) and (7) will be improved by knowledge of the linear system discrete time solution technique.

Preventive Maintenance of The Component

In the previous state transition diagram of the component, Figure 1., the transitions caused by preventive maintenance was not included. The state transition diagram with preventive maintenance is given in Figure 2. This shows that there is a transition from state 2 to state 1 with the rate of μ, provided by maintenance activity. The term u(t) is the step input function which is equal to 1 during the maintenance and zero otherwise.

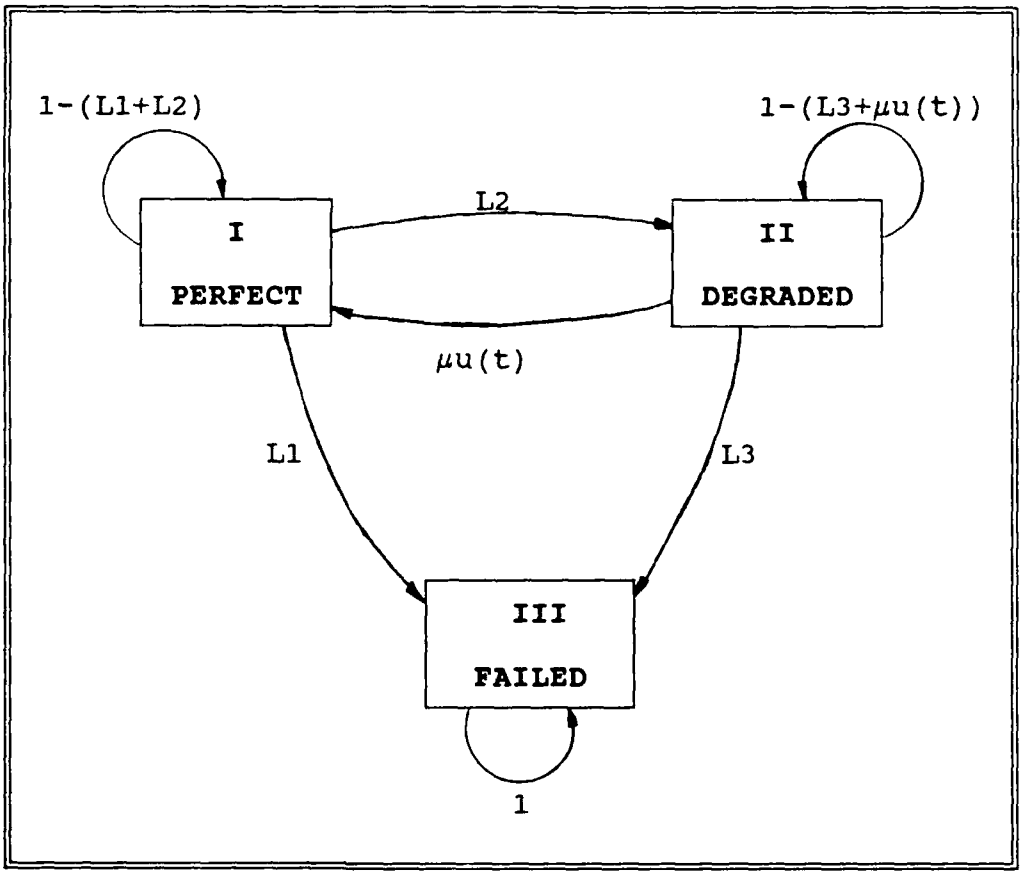

Figure 2. Component's state transition diagram under preventive maintenance.

According to that transition diagram the new transition matrix becomes:

$$\begin{bmatrix} 1-(L1+L2) & L2 & L1 \\ \mu u(t) & 1-(L3+\mu u(t)) & L3 \\ 0 & 0 & 1 \end{bmatrix} \qquad (23)$$

Note that it is not a simple transition matrix any longer. To simplify the process we will use two transition matrices: one representing the normal operation of component and the other representing the maintenance activity. The state transition diagram and transition matrix of the component during normal operation are the same as given in Figure 1 and Eq. 1 respectively. The transition matrix to represent the maintenance activity will be called as matrix $\underline{M}$. State transition diagram of component during the maintenance activity is given in Figure 3 and the matrix $\underline{M}$ is given in Eq. (24). During the maintenance there is no transition from state 1 to either state 2 or state 3. Only possible transitions are to stay in the current state or upgrade from state 2 to state 1.

$$\underline{M} = \begin{bmatrix} 1 & 0 & 0 \\ \mu & 1-\mu & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (24)$$

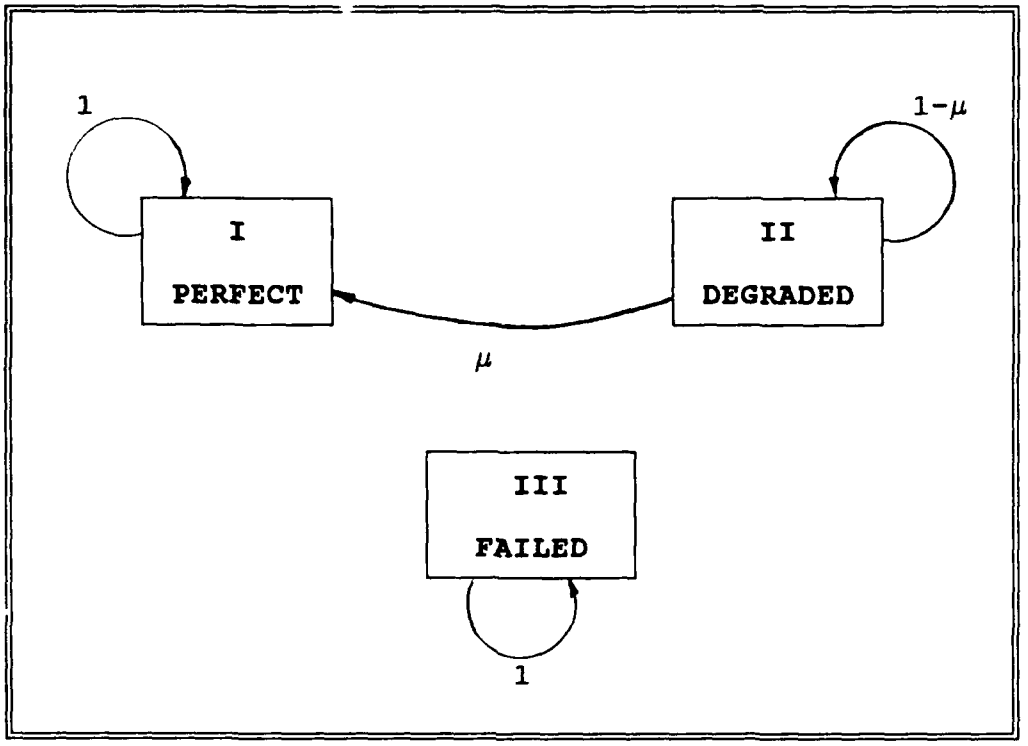

Figure 3. Component's state transition diagram during preventive maintenance.

If we rewrite the system differential equations given in Eq. (4) by using the new transition probabilities we have:

$$\dot{P}_1 = \frac{dP_1}{dt} = -(L1+L2)P_1(t) + \mu u(t)\ P_2(t)$$

$$\dot{P}_2 = \frac{dP_2}{dt} = (L2)P_1(t) - (L3 + \mu u(t))P_2(t)$$

$$\dot{P}_3 = \frac{dP_3}{dt} = (L1)P_1(t) + (L3)P_2(t) \qquad (25)$$

or in matrix notation;

$$\begin{bmatrix} \dot{P}_1 \\ \dot{P}_2 \\ \dot{P}_3 \end{bmatrix} = \begin{bmatrix} -(L1+L2) & 0 & 0 \\ L2 & -L3 & 0 \\ L1 & L3 & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \end{bmatrix} + \begin{bmatrix} 0 & \mu & 0 \\ 0 & -\mu & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \end{bmatrix} u(t)$$

or,

$$\dot{\underline{P}}(t) = (\underline{T} - I)^T \underline{P}(t) + (\underline{M} - I)^T \underline{U}(t) \qquad (26)$$

where $\underline{U}(t) = \underline{P}(t)u(t)$ and the reliability of component at a given time is again:

$$R(t) = P_1(t) + P_2(t) = [\ 1 \quad 1 \quad 0\ ]\ \underline{P}(t) \qquad (27)$$

Now, from equations (26) and (27) it is very easy to see the correspondence between our model and general linear state-space formulation given by Eq.(9) and Eq.(10). By defining A, B, C, Y(t), $\dot{X}(t)$ and X(t) of Eq.(9) and Eq.(10) as follows,

$$A = (\underline{T} - I)^T$$

$$B = (\underline{M} - I)^T$$

$$C = [\ 1 \quad 1 \quad 0\ ]$$

$$\dot{X}(t) = \dot{\underline{P}}(t)$$

$$X(t) = \underline{P}(t)$$

$$Y(t) = R(t)$$

The linear state-space formulation of component under preventivemaintenance becomes:

$$X(t) = AX(t) + BU(t) \tag{28}$$

$$Y(t) = CX(t) \tag{29}$$

Solutions to the (28) and (29) are given directly by equations (13) through (22). And now, to evaluate a components reliability with givens L1, L2, L3, μ, maintenance period and a certain discrete time step value (T), we only need to is to evaluate the equations (13) and (14) iteratively with a digital computer.

Before showing the implementation of the computer model with some examples, all the assumptions taken up to this point will be summarized in the next section.

Assumptions of The Single Component Model

1. Transition probabilities are a function of the length of the time interval in which transition occurs, not of the time of transition. Therefore it is a homogeneous, stationary Markov chain(1:112-141).

2. The transitional probabilities are not depend on the number of past transitions(1:112-141).

3. The probability of more than one transition in any interval is very small[9].

4. Maintenance action does not take any time or occurs at an instance without interrupting the normal operation.

5. Aging does not have any effect on the maintenance.

Program

Until this point, all the theory and the mathematical concept for single component model was developed. Now, the next step, is to translate the mathematical model into a computer program. In this study the simulation language SIMSCRIPT II.5_PC was used for all programming purposes. Its graphics capabilities provides a very good visual support.

The program accepts the input data either through the keyboard or through a data file(INPUT.DAT). Input data includes transition probabilities, L1, L2, L3, μ, maintenance period and total simulation time. The discrete time step, T has a fixed value of 0.1 seconds.

An initialization routine builds matrices A, B, and C and calculates A_T and B_T by using Eq.(21) and (22). Since all the elements in the transition matrices are less than one, power series of A_T with five terms expected to be reasonably accurate. Actually, it is as accurate as six digits after the decimal point; which we will see later in validation analysis. In case of perfect maintenance, $\mu = 1$, power series for B_T does not converge but, as we will see later it still gives acceptable accuracy. Therefore the equations that the program uses to calculate A_T and B_T become:

$$A_T = I + AT + \frac{A^2T^2}{2!} + \frac{A^3T^3}{3!} + \frac{A^4T^4}{4!}$$

$$B_T = B \left\{ IT + \frac{AT^2}{2!} + \frac{A^2T^3}{3!} + \frac{A^3T^4}{4!} \right\}$$

where T is the discrete time steps value for simulation.

After calculating the new matrices initialization writes them into an output file called OUT.DAT. Then sets the screen up for graphical output and starts the simulation clock.

Three processes called, SYSTEM.OPERATION, MAINTENANCE and OUTPUT work continuously until the end of the simulation and update system variables in given periods.

SYSTEM.OPERATION cycles every T=0.1 seconds and updates matrix X_T by

$$X_T(t) = A_T X_T(t-1) + \text{INPUT.MATRIX}$$

and resets the INPUT.MATRIX to zero.

Process MAINTENANCE cycles every maintenance period seconds and calculates new INPUT.MATRIX as $B_T X_T(t-1)$.

OUTPUT takes the current X_T and calculates the new reliability by multiplying matrix C with it in every discrete time step T=0.1.

SIMSCRIPT's built-in "smart-graph" feature graphs the value of reliability versus simulation time automatically. This feature reduced the effort needed for programming and made this task easier.

Example Problem

In this section the application of the model will be presented by an example problem. The input values for this example problem are given in Table 3.

L1	0.005
L2	0.25
L3	0.1
μ	1

Table 3. Problem Inputs

The transition matrices for this problem are:

$$\underline{T} = \begin{bmatrix} .745 & .25 & .005 \\ 0 & .9 & .1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \underline{M} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And the matrices used in the model can be calculated as follows:

$$A = (\underline{T}-I)^T = \begin{bmatrix} -.255 & 0 & 0 \\ .25 & -.1 & 0 \\ .005 & .1 & 0 \end{bmatrix} \qquad B = (\underline{M}-I)^T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

and,

$$A_T = \begin{bmatrix} .9748224 & 0 & 0 \\ .0245604 & .9900498 & 0 \\ .0006172 & .0099502 & 1 \end{bmatrix} \qquad B_T = \begin{bmatrix} 0 & .0987358 & 0 \\ 0 & -.0982663 & 0 \\ 0 & -.0004694 & 0 \end{bmatrix}$$

We will run this problem with different maintenance periods for total simulation times of 10 and 40 seconds. The range of the reliability curves for different maintenance periods defines an area which can be called the preventive maintenance envelope. The upper bound for this envelope is the value found by continuous preventive maintenance. This is a theoretic situation, it means that the system is always under preventive maintenance carried out simultaneously with the normal operation. And it can be simulated by setting the maintenance period to the value of discrete time steps, T=0.1. The lower bound of the preventive maintenance envelope is defined by the value found in case of no maintenance. This situation can be simulated by simply setting the maintenance period to a higher value then the simulation time.

The value μ=1 represents the perfect maintenance. That is any component operating in state 2 will transition to state 1 after the maintenance with 100% probability.

In the next two pages Figure 4. and Figure 5. show the preventive maintenance envelope for simulation times 10 and 40 seconds. Figure 6 and Figure 7. show the reliability ranges for different maintenance periods in 10 and 40 seconds simulation times respectively.

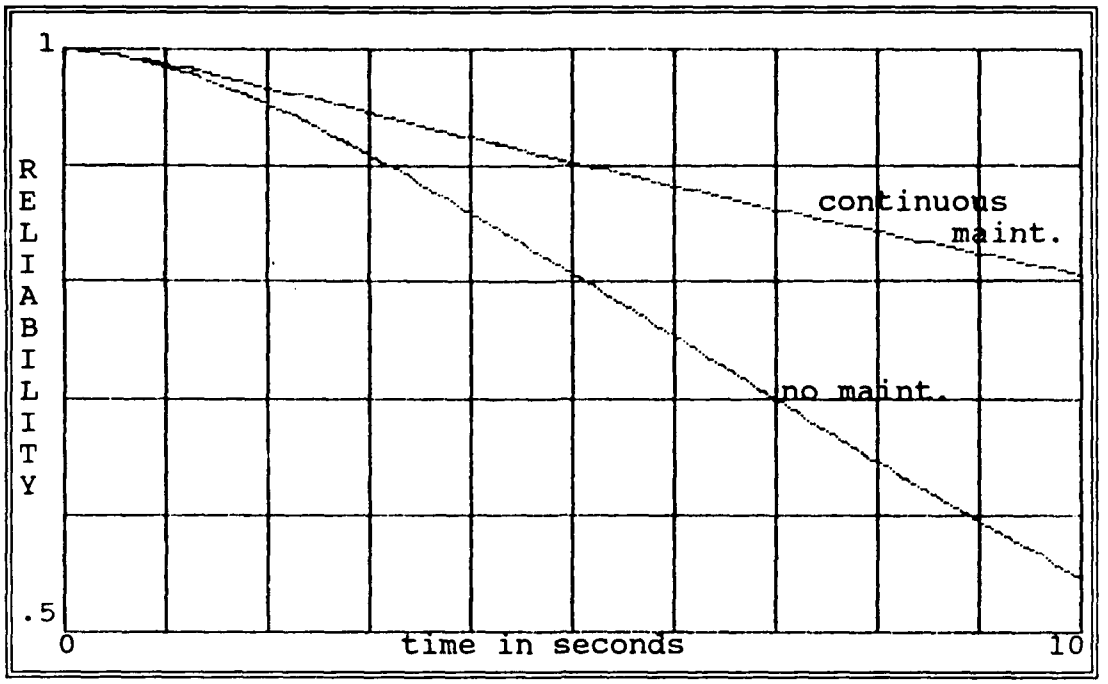

Figure 4. Reliability Envelope For 10 Seconds

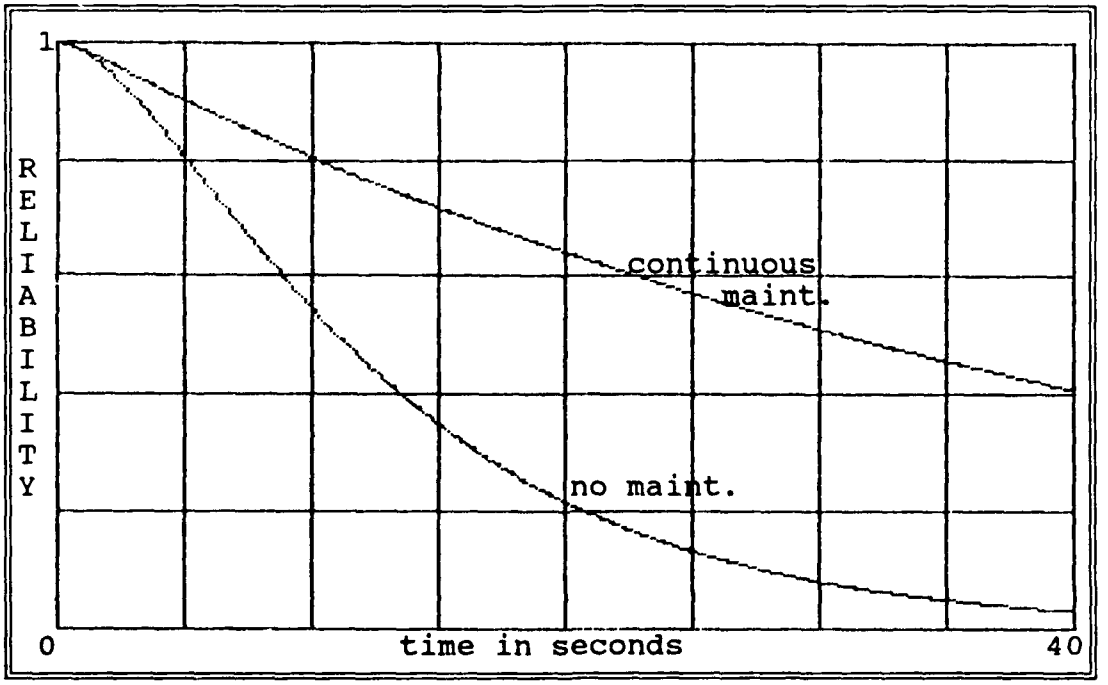

Figure 5. Reliability Envelope For 40 Seconds

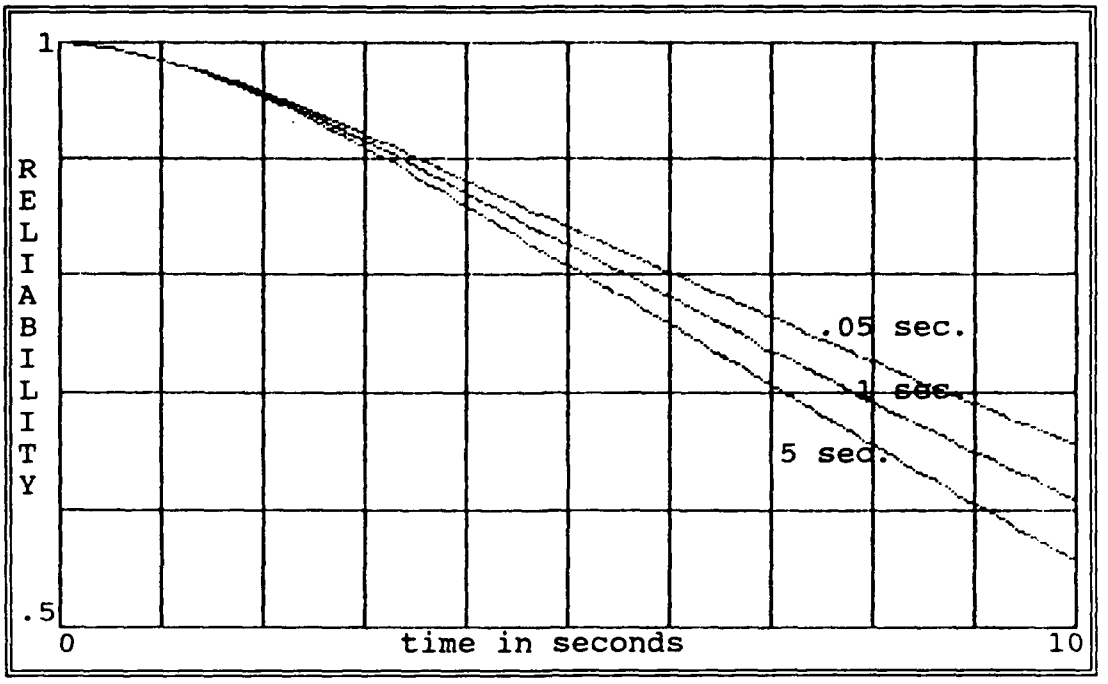

Figure 6. Reliability Ranges For 10 Seconds

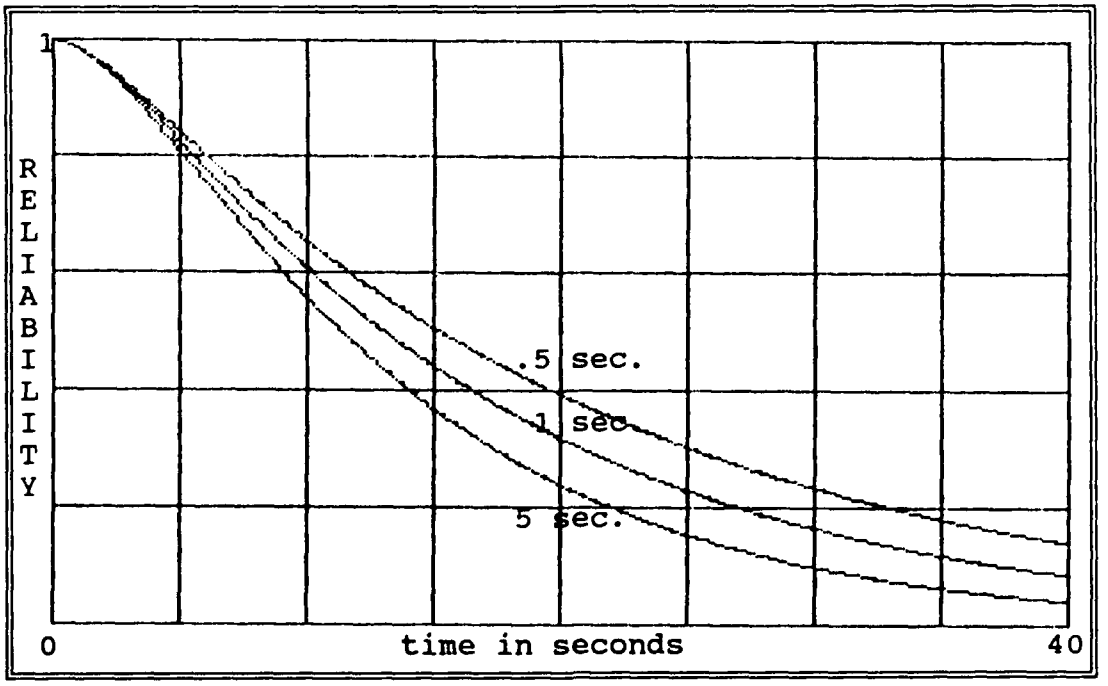

Figure 7. Reliability Ranges For 40 Seconds

Analysis of The Results And Model Validation

Before using the model in real problems we need to be sure whether it is yielding the right results. That is, a validation is needed. As pointed out earlier, the closed-form solution to the model is not always available. The equations in Eq.(25) can be solved for no maintenance(u(t) = 0 always) and continuous maintenance(u(t) = 1 always).

Now let's look at the case of no maintenance by setting u(t) = 0 :

$$\dot{P}_1(t) + (L1+L2)P_1(t) = 0 \qquad (31)$$

$$\dot{P}_2(t) - (L2)P_1(t) + (L3)P_2(t) = 0 \qquad (32)$$

These two equations are enough to find the system reliability because the system reliability is equal to the sum of P_1 and P_2. First we will solve the Eq.(31) by Laplace transformation method:

$$sP_1(s) - 1 + (L1+L2)P_1(s) = 0$$

$$P_1(s) = \frac{1}{s + (L1+L2)}$$

and by taking the inverse Laplace transform;

$$P_1(t) = e^{-(L1+L2)t} \qquad (33)$$

The Laplace transform for Eq.(32) is:

$$sP_2(s) + (L3)P_2(s) - \frac{L2}{s + (L1+L2)} = 0$$

$$P_2(s) = \frac{L2}{(s+L3)(s+L1+L2)}$$

and by taking the inverse Laplace transform;

$$P_2(t) = \frac{L2}{L1+L2-L3}\left[e^{-(L3)t} - e^{-(L1+L2)t} \right] \tag{34}$$

Now from Eq.(33) and Eq.(34), the reliability for no maintenance, $R_N(t)$ is :

$$\mathbf{R_N(t)} = P_1(t) + P_2(t) \tag{35}$$

$$= \frac{1}{L1+L2-L3}\left[(L2)e^{-(L3)t} - (L1-L3)e^{-(L1+L2)t} \right]$$

By setting u(t)=1 we can find the differential equations for <u>continuous</u> <u>maintenance</u>. Again the first two equations are enough to find the reliability:

$$\dot{P}_1(t) + (L1+L2)P_1(t) - \mu P_2(t) = 0 \tag{36}$$

$$\dot{P}_2(t) - (L2)P_1(t) + (L3+\mu)P_2(t) = 0 \tag{37}$$

The Laplace transformations for these equations are:

$$sP_1(s) + (L1+L2)P_1(s) - \mu P_2(s) = 1$$

$$sP_2(s) + (L3+\mu)P_2(s) - (L2)P_1(s) = 0$$

This equations form a linear system with 2 equations and two unknowns, and has to be solved simultaneously. Skipping the intermediate steps, the solution for $P_1(s)$ and $P_2(s)$ is:

$$P_1(s) = \frac{s + a}{(s-x_1)(s-x_2)} \; ; \quad P_2(s) = \frac{L2}{(s-x_1)(s-x_2)} \tag{38}$$

where;

$$\boxed{\begin{aligned} a &= L3 + \mu \\ b &= L1+L2+L3+\mu \\ c &= (L1+L2)(L3+\mu) - \mu L2 \\ x_{1,2} &= \frac{-b \pm \sqrt{b^2-4c}}{2} \end{aligned}} \tag{39}$$

And by taking inverse Laplace transform:

$$P_1(t) = \frac{(x_1+a)e^{x_1 t} - (x_2+a)e^{x_2 t}}{x_1 - x_2} \; ; \tag{40}$$

$$P_2(t) = \frac{e^{x_1 t} - e^{x_2 t}}{x_1 - x_2} L2 \tag{41}$$

From equations (40) and (41) the reliability function for continuous maintenance, $R_C(t)$ is as follows:

$$R_C(t) = \frac{(x_1+a+L2)e^{x_1 t} - (x_2+a+L2)e^{x_2 t}}{x_1 - x_2} \qquad (42)$$

By using $R_N(t)$ and $R_C(t)$, it is possible to check the values obtained by the computer model. Table 4. compares the values for no maintenance and shows the difference between exact solutions and the simulation results,

Time(sec.)	$R_N(t)$	Model	Difference
10	.54549740	.54549843	.00000103
20	.21454599	.21454655	.00000056
30	.08000995	.08001002	.00000027
40	.02951860	.02951869	.00000009

Table 4. Validation values for no maintenance case

As it is seen from the table, the model gives at least 6 digit accuracy . Comparison of the values in case of continuous maintenance is given in Table 5. Exact solutions are found by Eq.(42) and the model values are found by setting the maintenance period to discrete time step value, T=0.1 seconds.

The results are not as good as they were in no maintenance case. Negative difference values between exact solution and the model shows that the divergence of power series for B_T in case of $\mu = 1$ did not cause a significant problem.

Time(sec.)	$R_c(t)$	Model	Difference
10	.80623154	.80578482	- .00045672
20	.64124389	.64098889	- .00025500
30	.51001939	.50989592	- .00012347
40	.40564875	.40561384	- .00003491

Table 5. Validation values for continuous maintenance

To see the effect of μ, the same analysis is repeated for μ = .5. The difference between exact solution and model decreased to the half of the values which obtained in Table 5. Also the effect of the discrete time step value is studied. For T = .05, the difference decreased but resulted in considerable computational time. And the value T = .2 yielded larger differences.

Now we need to answer the question that "is the model dependable for the case of continuous maintenance?". First, the continuous maintenance is only a theoretical situation and second, the results only provide an upper bound. Third, let us try to make a real life analogy with the results of Table 5. If the component in this simulation is an aircraft engine and the discrete time steps represent a 2 hours flight then a 10 second simulation represents 200 flight hours while a 40 second simulation representing 800 flight hours. Clearly both simulation times are unnecessarily long to analyze the effect of preflight and post flight controls.

We can use the simulation times for 50 hour and 100 hour periodic maintenance analysis respectively. In this case we have chosen preventive maintenance periods of 2.5 seconds and 5 seconds instead of .1 seconds. The expected difference between the exact solution and the model becomes about $1.8x10^{-5}$ and $7.1x10^{-7}$. This indicates that in an actual problem we can expect at least 4 digit accuracy. More accurate results can be obtained by using longer simulation times in conjunction with a shorter time increment T. These are very good results, without resorting to the exact solution of a system of differential equations(which can be very difficult for multicomponent systems). The number of differential equations increase substantially as the number of system states increase. For each component, another set of equations need to be solved. But with this model, the only thing needed to analyze a system is the system parameters.

In the next chapter, the model will be extended to the multicomponent systems. The methods to construct the system matrices for multicomponent systems will be explained.

VI. Model For Multicomponent Systems

How To Interpret The Systems And How To Build System Matrices.

The linear state-space model for single component was developed in Chapter III. And analysis of the simulation results showed that the model is very efficient, fast and reasonably accurate compared to the difficult, time consuming solution of differential equations. The goal of this chapter is to find a generic way to build system matrices(A, B, C), for multicomponent systems. First, three simple systems will be studied. The methods to generate the system matrices will then be deduced from the results of these systems.

In the case of multicomponent systems, one additional assumption is necessary. It will be assumed that the operating condition of one component does not effect another component's operation. In other words, effect of the overloading will be ignored in case of one of the legs of a parallel system failed.

As in the last chapter, every component in the system will have three states, perfect, degraded and failed. The same terminology will be used for transition probabilities with one additional piece of notation to identify the component number. For example, transition probability from perfect to degraded for i^{th} component is Li1.

Two Components in Parallel

The system studied in this section has two components in parallel. Each component has three states and corresponding transition probabilities. Transition probabilities for component 1 are L11, L12, L13 and $\mu 1$. And the transition probabilities for component 2 are L21, L22, L23 and $\mu 2$. The system is pictured in Figure 8.

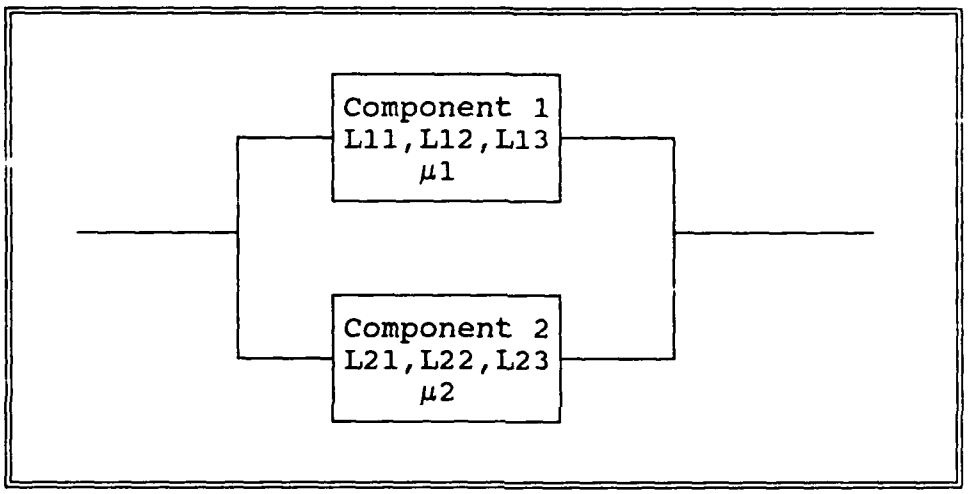

Figure 8. Two components in parallel.

To find the system states we need to define all possible operating conditions of the system. By using combinations of the states of component 1 and component 2, system states can be determined and are listed in Table 6. P, D and F indicate that the corresponding component is in a perfect, degraded or failed condition. The probabilities in system state transition matrices should be the multiplication of components' transition probabilities. Because each of them will represent a joint probability.

For example, when both components are in perfect condition or the system is in state 1, the probability of transition to state 9(both components failed) is L11xL21.

System states	Condition of component 1	Condition of component 2
1	P	P
2	P	D
3	P	F
4	D	P
5	D	D
6	D	F
7	F	P
8	F	D
9	F	F

Table 6. System states for two component system.

Before constructing the system transition matrices, let us simplify the notation somewhat. $\underline{T}1$ and $\underline{T}2$ will represent the transition matrices of component 1 and component 2 during operation. $\underline{M}1$ and $\underline{M}2$ will represent the transition matrices of component 1 and component 2 for maintenance:

$$\underline{T}1 = \begin{bmatrix} a1 & a2 & a3 \\ 0 & a4 & a5 \\ 0 & 0 & 1 \end{bmatrix} \qquad \underline{M}1 = \begin{bmatrix} 1 & 0 & 0 \\ a6 & a7 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{43}$$

$$\underline{T}2 = \begin{bmatrix} b1 & b2 & b3 \\ 0 & b4 & b5 \\ 0 & 0 & 1 \end{bmatrix} \qquad \underline{M}2 = \begin{bmatrix} 1 & 0 & 0 \\ b6 & b7 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{44}$$

The definitions for the terms in matrices $\underline{M}1$, $\underline{M}2$, $\underline{T}1$, $\underline{T}2$ are as follows:

a1 = 1-(L11+L12), a2 = L12, a3 = L11, a4 = 1-L13, a6 = μ1

a7 = 1-μ1 and

b1 = 1-(L21+L22), b2 = L22, b3 = L21, b4 = 1-L23, b6 = μ2

b7 = 1-μ2.

Now the system transition matrix during normal operation is as follows:

$$\underline{T} = \begin{bmatrix} a1b1 & a1b2 & a1b3 & a2b1 & a2b2 & a2b3 & a3b1 & a3b2 & a3b3 \\ 0 & a1b4 & a1b5 & 0 & a2b4 & a2b5 & 0 & a3b4 & a3b5 \\ 0 & 0 & a1 & 0 & 0 & a2 & 0 & 0 & a3 \\ 0 & 0 & 0 & a4b1 & a4b2 & a4b3 & a5b1 & a5b2 & a5b3 \\ 0 & 0 & 0 & 0 & a4b4 & a4b5 & 0 & a5b4 & a5b5 \\ 0 & 0 & 0 & 0 & 0 & a4 & 0 & 0 & a5 \\ 0 & 0 & 0 & 0 & 0 & 0 & b1 & b2 & b3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b4 & b5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $\underline{T}$ is an upper triangular matrix because there is no transition to perfect condition from any other state and there is no transition from failed states to degraded states. The only possible change in system's operational condition are to degrade, fail or remain in the same condition.

During the maintenance action, the only possible transition is from degraded state to perfect state, otherwise the system remains in its previous condition. The state transition diagram for maintenance is given in Figure 9. Every box represents a system state.

Numbers inside the boxes shows the state number as they are assigned in Table 6. And the letters in the boxes shows the conditions of component 1 and component 2 respectively.

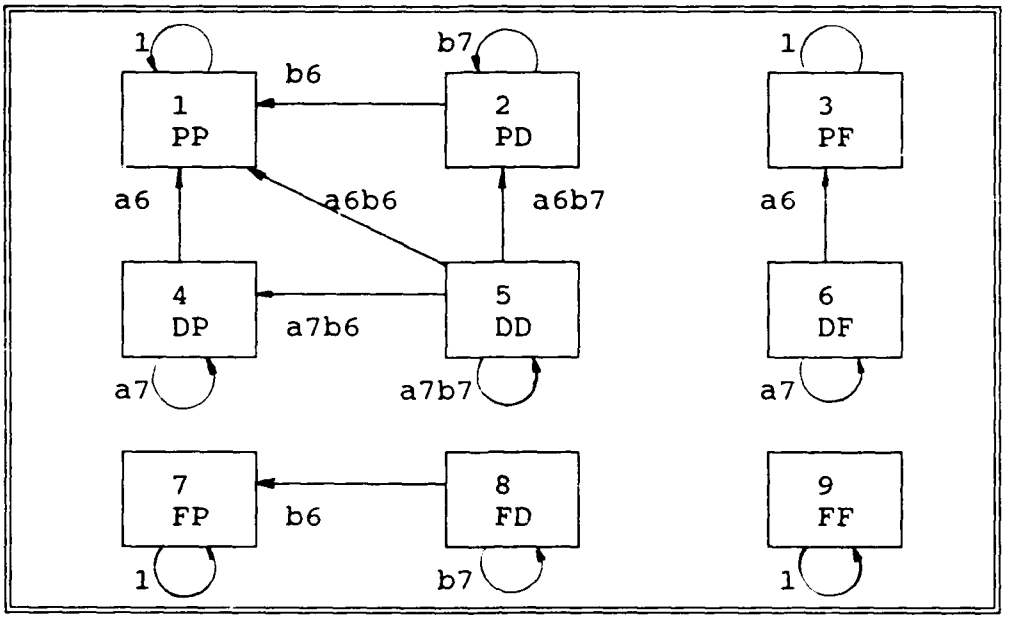

Figure 9. System transition diagram for maintenance of two component system.

Now, by using the transition probabilities in matrices $\underline{M}1$ and $\underline{M}2$ or directly using the numbers in Figure 9 the system transition matrix for maintenance is as follows:

$$\underline{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b6 & b7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a6 & 0 & 0 & a7 & 0 & 0 & 0 & 0 & 0 \\ a6b6 & a6b7 & 0 & a7b6 & a7b7 & 0 & 0 & 0 & 0 \\ 0 & 0 & a6 & 0 & 0 & a7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & b6 & b7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $\underline{M}$ is a lower triangular matrix because it represents only the system upgrade.

Now we are ready to write the first equation of the linear, state-space formulation:

$$\dot{X} = (\underline{T} - I)^T X(t) + (\underline{M} - I)^T U(t)$$
$$= AX(t) + BU(t) \qquad (45)$$

where $U(t) = X(t)$, when t is a maintenance period, and
$= 0$, otherwise.

And both U(t) and X(t) are 9x1 column matrices. The equation (45) is the same as equation (28) which is derived for single component.

To find the system output, the reliability, we need to define the C matrix. Since, two component are in parallel, the system is operative if any one of the components is operative. This means that the system is operative in every state except state 9 which both components are failed. Therefore, to find the system reliability, we need to find the sum of the probabilities of being in first eight states. And the C matrix should be a 1x9 row matrix as follows:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

and the system output is:

$$Y = CX(t) \qquad (46)$$

Again the equation (46) is the same as equation (29) which is derived for single components output.

Two Components in Series

This system also has two components, but the components are connected in series. This system is pictured in Figure 10.

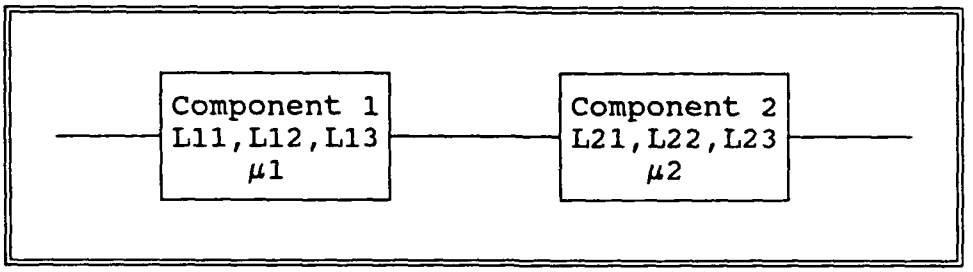

Figure 10. Two component in series.

Again the two components have the same transition probabilities as assigned in the first system and in Figure 10.

Clearly, changing the connection type does not change the system states. The same system states defined in Table 6 can be used for this system too. And by using the same transition matrices in (43) and (44) for components, the system transition matrices become the same as they were in the parallel system. Equation (45) is also valid for this system.

Because the only difference between two systems is the connection type, the formulation for two systems will differ only in the output. The serial system is operative only if the both components are operative. That is, when it is in states 1, 2, 4, or 5. Therefore the C matrix in the serial case will have ones as its first, second, fourth and fifth elements and zeros for all others:

$$C = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A Three Component System

This system combines the first two models. It has two components in parallel and the third one serial to the first two. This system is pictured in Figure 11.

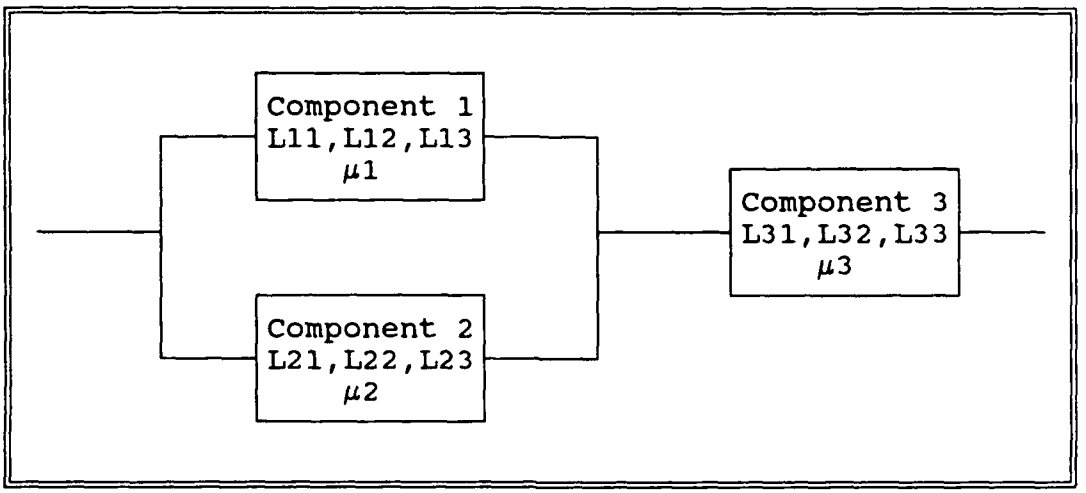

Figure 11. A three component system.

To find the system states, we need to look at the every possible combinations of all three components' operating conditions. And this will give 27 system states as in the classical Markov modelling. This states are listed in Table 7 where P, D and F represents the corresponding component's condition.

The system states in Table 7 can be represented in more understandable way. Instead of listing from 1 to 27, let us define a three component state vector. Which every element in the vector represents one component's condition and can

have the value of 1, 2 or 3 corresponding to perfect, degraded and failed states. For example, state 1 which all three components are in perfect condition can be represented as <1,1,1>, or state 27 which all components are inoperative can be represented as <3,3,3>. Clearly combinations of three numbers in 3 dimensional vector generates 27 different vectors. And if we arrange this state vectors as three digit numbers from smaller to bigger, then the same order of states in Table 7 can be produced.

System states	Condition of component 1	Condition of component 2	Condition of component 3
1	P	P	P
2	P	P	D
3	P	P	F
4	P	D	P
5	P	D	D
6	P	D	F
7	P	F	P
8	P	F	D
9	P	F	F
10	D	P	P
11	D	P	D
12	D	P	F
13	D	D	P
14	D	D	D
15	D	D	F
16	D	F	P
17	D	F	D
18	D	F	F
19	F	P	P
20	F	P	D
21	F	P	F
22	F	D	P
23	F	D	D
24	F	D	F
25	F	F	P
26	F	F	D
27	F	F	F

Table 7. System states for three component system.

Two 27x27 matrices can be generated as matrix M and matrix T in the same manner as the first two systems. For example, in matrix M the transition probability from state <1,1,1> to <3,3,3> can be found by multiplication of each three components' transition probabilities from state 1(perfect) to state3(failed): L11xL21xL31.

For matrix C, the entry for state <i,j,k> is 1 if either i or j is not equal to 3 and k is 1 or 2. That is at least one of the parallel components is operative and component 3 is operative. In all other cases the entry in matrix C will be zero. Matrix C should be as follows:

$$C = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

By using the state vector idea in the next section, a general method will be presented for building the system matrices.

How To Build System Matrices

To be able to use the same program for any system we need to find a method that by using this method the system matrices can be generated by the computer. Otherwise for large complex systems, building the matrices of the system would take too much time. To improve our program in this way, we need to define a method to interpret the multicomponent systems with minimum amount of data.

To interpret the systems we could think of them a collection of the simple components and connections between the components. In first two systems studied above, component 1 and component 2 may represent two multicomponent systems connected to each other in parallel or in series. And in this case the system transition matrices $\underline{M}$ and $\underline{T}$ can be generated in the same way by using the binary state vector combinations. Only difference will be the size of the matrices. For example, let us look at the parallel connection of the first two systems as it is pictured in Figure12.

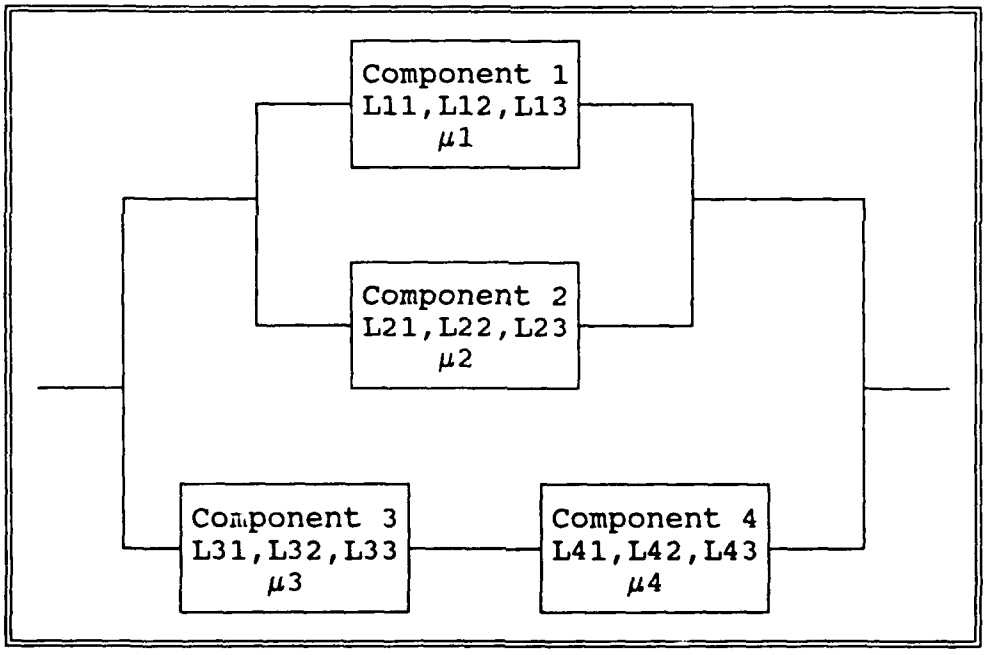

Figure 12. Two two component systems in parallel.

Now if we list the two component state vectors which both entries in the state vector has the range of 1 to 9 we can generate all the system states and calculate the system transition matrices $\underline{M}$ and $\underline{T}$.

For the C matrix we need to use the information about the connection type. For example the entry for state <3,2> in the C matrix should be 1, because the upper part of the system is operative(Component 1 is in perfect state and component 2 is in failed state) and also the lower part of the system is operative(component 3 is in perfect state and component 4 is in degraded state). This result can be found faster by using the C matrices of each part. Because the 3^{rd} entry of the C matrix of upper part is 1 which means operative and the 2^{nd} entry of the C matrix of lower part is also 1. If we look at the states <9,3>, <9,6>, <9,7>, <9,8> or <9,9> the entry in the C matrix should be 0, because both parts in a parallel connection are inoperative. This could be seen from the individual C matrices, because the 9^{th} entry of upper part's C matrix and 3^{rd}, 6^{th}, 7^{th}, 8^{th}, 9^{th} entries of the lower part's C matrix are all zero indicating that in this states both parts are inoperative.

In order to interpret a system which has more than one simple component, the smallest dimension of the state vector is 2. And by choosing a proper range for the entries in the state vector, any system can be interpreted by two dimension state vectors. For example if we add a 5^{th} component serial to the system in Figure 12, the first entry in the state vector will have the range of 81, and, since the new part is a simple component the second entry will have the range of 3. Combination of these numbers will generate 243($=3^5$) states of a 5 component system.

This examples showed that any system can be analyzed by evaluating its parts as couples connected to each other in parallel or in series. By starting from simple components the whole system can be analyzed step by step. Now we need to define the term couple:

Definition: A couple is a subsystem which is composed of two parts where the two parts may be simple components or some other couples. Simple components have three operating states. And every couple has a connection type of either parallel or serial.

By using this definition the system in Figure 12 has 3 couples. The first couple is component 1 and component 2 and the connection type is parallel. The second couple is component 3 and component 4 and the connection type is serial. And the third couple is the couple composed of couple 1 and couple 2 and its connection type is parallel.

If the first part in a couple has m states and the second part has n states then couple will have mxn states. And the first entry in state vector will have the range of m while the second entry has the range of n.

Every couple has a matrix M, matrix T and matrix C. To find the elements of that matrices let T1, M1, C1 be the matrices of part one and T2, M2, C2 be the matrices of part two. The transition rates from state <i,j> to <k,l> or the elements in the intersection of row <i,j> and column <k,l> of the matrices T and M can be found through the following equations:

$$\underline{T}(<i,j>,<k,l>) = \underline{T1}(<i,k>) \times \underline{T2}(<j,l>) \quad (47)$$

$$\underline{M}(<i,j>,<k,l>) = \underline{M1}(<i,k>) \times \underline{M2}(<j,l>) \quad (48)$$

and

$$C(<i,j>) = \begin{cases} 1, & \text{if } C1(<i>) = C2(<j>) = 1 \\ 1, & \text{if } C1(<i>) + C2(<j>) = 1 \text{ and connection type is parallel} \\ 0, & \text{otherwise.} \end{cases} \quad (49)$$

To illustrate these equations in an example, let us rename the 9 states of two component system, given in Table 6, from <1,1> to <3,3> and use the same transition probabilities in matrices (43) and (44). Then the transition probability from state <2,1> to state <2,3>(from degraded perfect to degraded failed) during the normal operation is (1-L13)*L21 and would be calculated with Eq. (47) as:

$\underline{T}(<2,1>,<2,3>) = \underline{T1}(<2,2>) \times \underline{T2}(<1,3>) = a4b3 = (1-L13)*L21$

which is the same as the 6^{th} element of the 4^{th} row of $\underline{T}$ matrix in page 43. The transition probability during the maintenance action should be zero, because the second component can not fail while the system is under maintenance. The same result can be found with Eq. (48) as:

$\underline{M}(<2,1>,<2,3>) = \underline{M1}(<2,2>) \times \underline{M2}(<1,3>) = (a7)(0) = 0$

which is the same as the 6^{th} element of the 4^{th} row of $\underline{M}$ matrix in page 44. The entry in the C matrix of the parallel couple for state <2,3> should be 1, because the first component is still operative. The same result can be obtained with Eq.(49), because C1(<2>)+C2(<3>) = 1+0 = 1 and connection type is parallel.

Now, the next thing we need to do is to add an additional routine to the previous program which builds the system matrices according to the rules (47), (48), and (49).

Modifications on The Program

The program treats every component and every couple as a permanent entity. Attributes which define every component are its transition probabilities, L1, L2, L3, and μ. In the first program components were represented directly by their transition matrices but for multicomponent models it takes too much storage. A routine called EXTRACT.MATRICES builds the component's matrices only when they are needed. After they are used, the program releases all component matrices.

Every couple has eight attributes called Part one, Part two, Number one, Number two, Connection type, A matrix, B matrix, and C matrix. The new routine called BUILD.MATRIXES starts with first simplest couple(would be a simple component) and builds the transition matrices step by step until the last couple, the system itself.

This routine is basicly four nested "for loop" which starts with the first row of the first part's transition matrices and takes the combinations of the elements of this row with the first row of the second part's transition matrices by order to find the first row of the couple's A matrix or B matrix.

This continues with the other rows of the second part's matrices to find all combinations for the first row of the first part's matrices. The same process continues for each row in the first part's matrices. For C matrix a variable called SCALE is set to one for serial connection and zero otherwise. This variable is then subtracted from the sum of the entries in the both parts' C matrixes for every combination. If the result is bigger then zero then the entry in the new C matrix is set to 1, otherwise it is set to zero. If the connection is parallel, the result is zero if and only if the both numbers from two matrices are zero(both parts are inoperative). If the connection is serial, the result is one if and only if both numbers from two matrices are one(both parts are operative). Once a couple's matrices are built, the old matrices belonging to the parts of the couple are released so that the memory usage is dynamically minimized.

To ease data entry two new editing menu were added to the program. One menu is used for editing a specific component's parameters and the other menu is used for editing a specific couple's parameters.

Both menus can be pulled down any time before or after the simulation. This gives the ability to run the same system with slight changes without the need to enter all the system parameters again or to leave the program for a few changes in the input file.

Another routine called RECORDER was also added to the program. This routine records the system output(Reliability) for given time periods. In this way the output of a run can be saved for later analysis.

Input File Format

The input file has three paragraphs. First paragraph is used for components, the second paragraph is used for couples, and the last is used for the maintenance period, recording period, and simulation time.

The first line of the first paragraph is the number of components and the each following line gives the components' parameters(L1, L2, L3, μ) by order.

The first line of the second paragraph is the number of couples and the each following line gives the couples' parameters(Part one, Part two, Connection type) by order.

```
4
L11   L12   L13   μ1
L21   L22   L23   μ2
L31   L32   L33   μ3
L41   L42   L43   μ4

3
PARALLEL  COMPONENT 1 COMPONENT 2
SERIAL  COMPONENT 3  COMPONENT 4
PARALLEL  COUPLE 1  COUPLE 2

MAINTENANCE.PERIOD
RECORDING.PERIOD
SIMULATION.TIME
```

Figure 13. An example input file.

The last paragraph consists of three lines: maintenance period, recording period, and simulation time in seconds.

The input file for the system in Figure 12 should look like Figure 13.

Example Outputs

To illustrate the implication of the last modifications let us run the program for the parallel and serial couples given in Figure 8 and Figure 10. The same parameters given in Table 3 will be used for both parts in the couples. System matrices except matrix C are the same for both parallel and serial couple:

$$A = (\underline{T}-I)^T = \begin{bmatrix} -.445 & .0 & .0 & .0 & .0 & .0 & .0 & .0 & .0 \\ .186 & -.330 & .0 & .0 & .0 & .0 & .0 & .0 & .0 \\ .004 & .075 & -.255 & .0 & .0 & .0 & .0 & .0 & .0 \\ .186 & .0 & .0 & -.330 & .0 & .0 & .0 & .0 & .0 \\ .062 & .225 & .0 & .225 & -.190 & .0 & .0 & .0 & .0 \\ .001 & .025 & .250 & .004 & .090 & -.100 & .0 & .0 & .0 \\ .004 & .0 & .0 & .075 & .0 & .0 & -.255 & .0 & .0 \\ .001 & .004 & .0 & .025 & .090 & .0 & .250 & -.10 & .0 \\ .000 & .001 & .005 & .001 & .010 & .100 & .005 & .100 & .0 \end{bmatrix}$$

$$B = (\underline{M}-I)^T = \begin{bmatrix} 0. & 1. & 0. & 1. & 1. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

$$A_T = \begin{bmatrix} .956 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ .018 & .968 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ .0004 & .007 & .975 & 0. & 0. & 0. & 0. & 0. & 0. \\ .018 & 0. & 0. & .968 & 0. & 0. & 0. & 0. & 0. \\ .006 & .022 & 0. & .022 & .981 & 0. & 0. & 0. & 0. \\ .0002 & .003 & .025 & .0005 & .009 & .990 & 0. & 0. & 0. \\ .0004 & 0. & 0. & .007 & 0. & 0. & .975 & 0. & 0. \\ .0002 & .0005 & 0. & .003 & .009 & 0. & .025 & .990 & 0. \\ .000 & .0001 & .001 & .0001 & .001 & .01 & .001 & .01 & 1. \end{bmatrix}$$

$$B_T = \begin{bmatrix} 0. & .098 & 0. & .098 & .098 & 0. & 0. & 0. & 0. \\ 0. & -.097 & 0. & .00[illegible] & .001 & 0. & 0. & 0. & 0. \\ 0. & -.0003 & 0. & .000 & .000 & .099 & 0. & 0. & 0. \\ 0. & .001 & 0. & -.097 & .001 & 0. & 0. & 0. & 0. \\ 0. & -.001 & 0. & -.001 & -.099 & 0. & 0. & 0. & 0. \\ 0. & -.0001 & 0. & -.000 & -.0004 & -.098 & 0. & 0. & 0. \\ 0. & .000 & 0. & -.0003 & .000 & 0. & 0. & .099 & 0. \\ 0. & -.000 & 0. & -.0001 & -.0004 & 0. & 0. & -.098 & 0. \\ 0. & -.000 & 0. & -.000 & -.0001 & -.0005 & 0. & -.0005 & 0. \end{bmatrix}$$

As given before, the C matrix for a parallel couple is:

$$C_{PARALLEL} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

And the C matrix for a serial couple is :

$$C_{SERIAL} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Reliability range plots for both couples are given in Figure 14 and Figure 15.

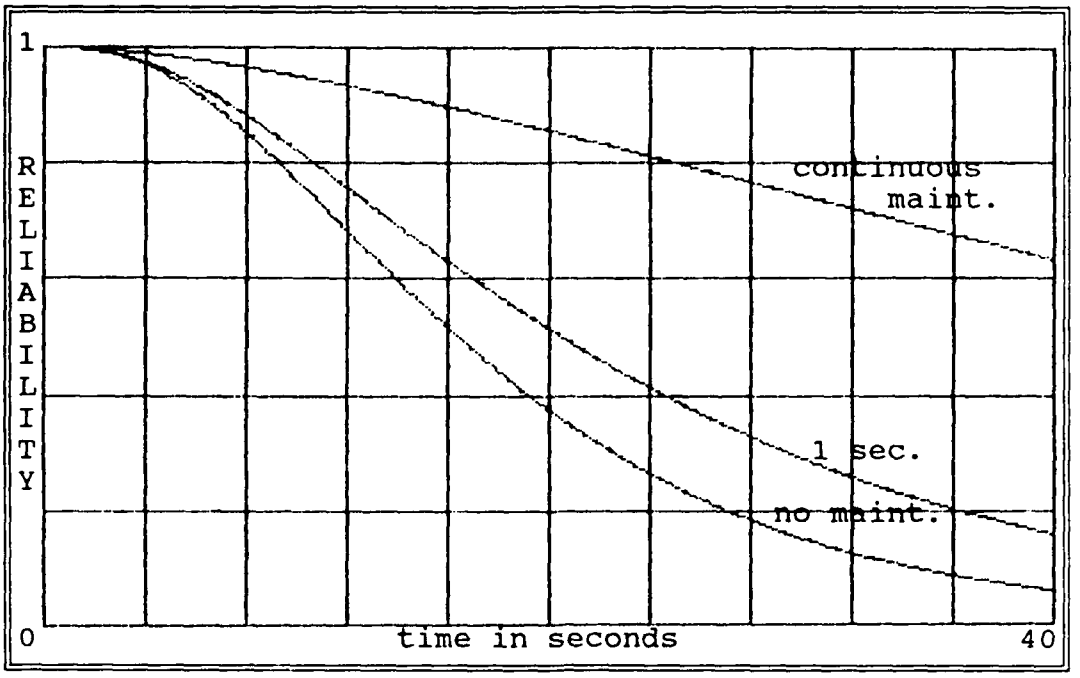

Figure 14. Reliability Ranges for Parallel Couple

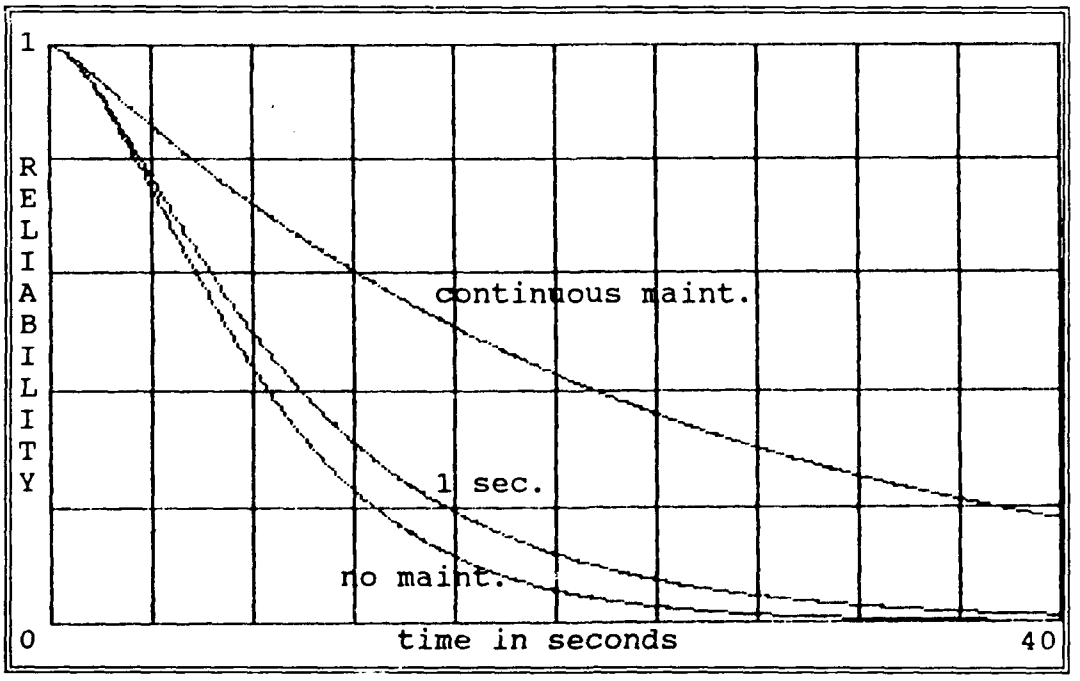

Figure 15. Reliability Ranges for Serial Couple

Now the model is complete to use for the analysis of any system. Only constraint on the capability of the program is the memory and the speed of the particular machine on which the program is used.

The summary of the whole research will be given in the last chapter. Also the possible suggested areas for further researchs will be addressed.

V. Summary And Conclusions

Summary of the Research

The main purpose of this research was to develop a model of scheduled preventive maintenance that can be used for system reliability predictions. The most popular method among previously used modeling techniques is the Markov modeling technique. And the motivation for the current research came from the state growth problem of the Markov modeling. The total number of states in a Markov model grows exponentially as the number of components in the system increase; the method becomes inefficient for large systems. The methodology chosen for this research was to use the linear state-space formulation on the Markov model of the systems under preventive maintenance.

To simplify the problem, the modeling was done initially on a single component system. After defining the Markov states and deriving the difference equations for the single component under preventive maintenance, the theory behind the linear, state-space formulation was introduced. The linear, state-space formulation was then applied to the model of the single component by using a discrete time solution.

The research objectives were to develop an easy-to-use computer model, which approximates the exact solution of the Markov model in a reasonable accuracy, runs fast, requires a minimum amount of data and provides low operating cost. The results of the first program written for the single component system showed that the model meets the all objectives with the accuracy of at least 10^{-5} on the system reliability prediction.

To expand the model to the multicomponent systems, a method to build the system matrices was sought in the Chapter IV. And the idea of couples was introduced as a tool to interpret the multicomponent systems and build the system matrices iteratively. The process to build the system matrices starts with the first single component and calculates the combinations of the states of all components step by step. With this method it is possible to run the model with a minimum amount of data which includes only the transition probabilities for all components and the connection types between them.

Conclusions of the Research Effort

The objectives of this research were met. A model which is easy-to-use, which runs fast, and requires a minimum amount of data was developed. The user friendly features of the program are:

- Initial input data can be entered in both ways; an input data file or directly the keyboard.

- The reliability plot is provided during the simulation as well as in an output data file in which the system reliability is recorded for desired time periods.

- The input menus also allow for sensitivity analysis, the the same system may be analyzed again with slight changes.

The program is very efficient compared to the effort required by classical Markov modeling. The main reason for the efficiency of the model is that the model needs no extensive programming or modeling effort. Any system can be analyzed with the minimum amount of data.

The program provides low operating cost. It runs on any personal computer and saves modeling time. Sensitivity analyses can be done very fast without additional preparations. The input data is saved until the end of the simulation; and if the user wants to continue with the same system, the program introduces the input menus and reruns the same system with the new set of parameters.

The results produced by the program are very dependable. Validation studies in Chapter III showed that the results are accurate to at least five digits.

The input data includes the transition probabilities of each component and the connection types between the components. The maintenance period, simulation time and, if desired, the recording period are also needed as inputs.

Suggested Areas of Further Research

The further research areas can be directed in two ways. One is to improve the program which was developed in this research. And the second is to accompany the model with an objective function and add an optimization task to the program.

In this research, Simscript II.5_PC was selected as the programming language. The current program can be improved in two ways:

- The same program may be modified to run more efficiently with the same language.

- Or another language can be used. Simscript has some very good features but it may not be the fastest language for the purpose.

The program is already very useful for the "what-if" type studies of the design engineers during the early design process. But a logistics planner also wants to know the operating cost of the system under suggested maintenance action. In other words he needs to predict the expenses of the maintenance and more than that needs to optimize the useful life of the system under the budget constraints. Therefore the most important further research area is to accompany this model with an optimization problem. Two dual problems can be introduced; maximizing the system reliability for a given maintenance budget or minimizing the maintenance expenses for a required system reliability.

APPENDIX

Computer Program Source Code

This program written in Simscript II.5_PC to run on any IBM PC compatible computer. Program accepts the input data either from a data file(INPUT.DAT) or directly from the keyboard. Outputs include a run time graphics screen, an output file for system matrices(OUT.DAT) and another output file(RECORD.DAT) to record system reliability in given time periods.

To run the program a graphics file(TRACE.GRF) is needed which can be easily generated by using Simscript's graphics editor.

```
preamble

define INDEX and ERROR.FLAG as integer variables
define T,MAINT.PERIOD, REC.PERIOD and SIM.TIME
                                        as real variables
define ERROR.MESSAGE as a text variable
define  MATRIX.A,
        MATRIX.B,
        MATRIX.C,
              AT,
              BT,
              UT,
              XT,
               E
           and INPUT.MATRIX   as 2-dim,real arrays

permanent entities
  every COMPONENT has
          an L1,
          an L2,
          an L3
       and a Mu
define L1,L2,L3,Mu as real variables

  every COUPLE has
           a CONNECTION.TYPE,
           a PART.ONE,
           a PART.TWO,
           a NUMBER.ONE,
           a NUMBER.TWO,
          an A.MATRIX,
           a B.MATRIX
       and a C.MATRIX
 define CONNECTION.TYPE, PART.ONE and PART.TWO as text
                                                  variables

define A.MATRIX,B.MATRIX,C.MATRIX, NUMBER.ONE and NUMBER.TWO
                                        as integer variables

 processes include MAINTENANCE, SYSTEM.OPERATION, OUTPUT
                                              and RECORDER

display variables include RELIABILITY
define RELIABILITY as a real variable

end ''PREAMBLE

MAIN

  call INTRO.MESSAGE
  start simulation
end ''MAIN
```

```
routine BUILD.MATRIXES.OF(THIS.COUPLE)

 define THIS.COUPLE,I,J,K,L,M,N,P,R,S,SCALE as integer
                                              variables
 define A1,A2,B1,B2,C1,C2,NEW.A,NEW.B,NEW.C
                                        as 2-dim, real arrays

select case CONNECTION.TYPE(THIS.COUPLE)

case "PARALLEL" , "parallel", "P", "p"
   let SCALE = 0

case "SERIAL", "serial", "S", "s"
   let SCALE = 1
default
endselect

 if (PART.ONE(THIS.COUPLE) = "COUPLE") or
    (PART.ONE(THIS.COUPLE) = "couple")

    let A1(*,*) = A.MATRIX(NUMBER.ONE(THIS.COUPLE))
    let B1(*,*) = B.MATRIX(NUMBER.ONE(THIS.COUPLE))
    let C1(*,*) = C.MATRIX(NUMBER.ONE(THIS.COUPLE))

  else

   call EXTRACT.MATRIXES.OF(NUMBER.ONE(THIS.COUPLE))
                                        yielding A1(*,*),
                                                 B1(*,*)
                                             and C1(*,*)
 always


 if (PART.TWO(THIS.COUPLE) = "COUPLE") or
    (PART.TWO(THIS.COUPLE) = "couple")

    let A2(*,*) = A.MATRIX(NUMBER.TWO(THIS.COUPLE))
    let B2(*,*) = B.MATRIX(NUMBER.TWO(THIS.COUPLE))
    let C2(*,*) = C.MATRIX(NUMBER.TWO(THIS.COUPLE))



  else

   call EXTRACT.MATRIXES.OF(NUMBER.TWO(THIS.COUPLE))
                                        yielding A2(*,*),
                                                 B2(*,*)
                                           and C2(*,*)
 always
```

```
let N = dim.f(A1(*,*))
let M = dim.f(A2(*,*))
reserve NEW.A and NEW.B as (N*M) by (N*M)
reserve NEW.C as 1 by (N*M)


let P = 0                        ''Reset row # of NEW.A & NEW.B.
for I = 1 to N                   ''Index for rows of A1 or B1.
  do
   for K = 1 to M                ''Index for rows of A2 or B2.
    do
     let P = P+1               ''Current row # for NEW.A & NEW.B
     let S = C1(1,I)+C2(1,K)-SCALE  ''Calculate column P of
                                                        NEW.C.
     if S gt 0
       let NEW.C(1,P) = 1
      otherwise
       let NEW.C(1,P) = 0
     always

     let R = 0                ''Reset column # of NEW.A & NEW.B.

     for J = 1 to N           ''Index for columns of A1 or B1.
      do
       for L = 1 to M         ''Index for columns of A2 or B2.
        do
         let R = R+1    ''Current column # for NEW.A & NEW.B.

          let NEW.A(P,R) = A1(I,J)*A2(K,L)
          let NEW.B(P,R) = B1(I,J)*B2(K,L)

        loop
     loop
  loop
loop

let A.MATRIX(THIS.COUPLE) = NEW.A(*,*)
let B.MATRIX(THIS.COUPLE) = NEW.B(*,*)
let C.MATRIX(THIS.COUPLE) = NEW.C(*,*)

let MATRIX.A(*,*) = NEW.A(*,*)
let MATRIX.B(*,*) = NEW.B(*,*)
let MATRIX.C(*,*) = NEW.C(*,*)

release A1,A2,B1,B2,C1 and C2  ''Get rid of the old
matrixes.
end ''BUILD.MATRIXES
```

```
routine COMPONENT.MENU(N)

   define N as an integer variable
   define CHOICE as an alpha variable
   define ANSWER as a real variable


   call vbcolor.r(1)
   call vfcolor.r(15)
   let lines.v = 0

   while 0 = 0
    do
      call vclears.r
      print 10 lines with N, L1(N), L2(N), L3(N), Mu(N) thus


                              EDITING COMPONENT # *

                  L1,transition from perfect to failed   = *.***
                  L2,transition from perfect to degraded = *.***
                  L3,transition from degraded to failed  = *.***
                  Mu,repair rate                         = *.***


      print 10 lines thus


                              1) Change L1
                              2) Change L2
                              3) Change L3
                              4) Change Mu

                              D) Done with this component


   call vgotoxy.r(21,0)
   write as "Enter your choice => ", +
   call rcr.r
   read CHOICE as A 1
   call vgotoxy.r(21,0)
   call vclearl.r

   select case CHOICE

   case "1"
     write as "Enter new L1 => ", +
     read ANSWER
     let L1(N) = ANSWER
```

```
    case "2"
      write as "Enter new L2 => ", +
      read ANSWER
      let L2(N) = ANSWER

    case "3"
      write as "Enter new L3 => ", +
      read ANSWER
      let L3(N) = ANSWER

    case "4"
      write as "Enter new Mu => ", +
      read ANSWER
      let Mu(N) = ANSWER

    case "D", "d" , "E" , "e" , "Q" , "q" , "X" , "x"
      leave

      default
      endselect
  loop
end       ''COMPONENT.MENU
```

```
routine COUPLE.MENU(N)

   define N and NUMBER as an integer variable
   define CHOICE as an alpha variable
   define ANSWER as a text variables

   call vbcolor.r(1)
   call vfcolor.r(15)
   let lines.v = 0

   while 0 = 0
    do
      call vclears.r

      print 10 lines with N, CONNECTION.TYPE(N),
                          PART.ONE(N), NUMBER.ONE(N),
                          PART.TWO(N), NUMBER.TWO(N) thus


                              EDITING COUPLE # *

                          Connection type : ********
                          Part One        : *********
                          Number One      : **
                          Part Two        : *********
                          Number Two      : **


      print 10 lines thus

                          1) Change Connection Type
                          2) Change Part One
                          3) Change Number One
                          4) Change Part Two
                          5) Change Number Two

                          D) Done with this couple


   call vgotoxy.r(21,0)
   write as "Enter your choice => ", +
   call rcr.r
   read CHOICE as A 1
   call vgotoxy.r(21,0)
   call vclearl.r

   select case CHOICE

   case "1"
     write as "Enter new connection type => ", +
     read ANSWER
     let CONNECTION.TYPE(N) = ANSWER
```

```
    case "2"
       write as "What Is New Part One  => ", +
       read ANSWER
       let PART.ONE(N) = ANSWER

     case "3"
       write as "Enter new Number One => ", +
       read NUMBER
       let NUMBER.ONE(N) = NUMBER

     case "4"
       write as "What Is New Part Two => ", +
       read ANSWER
       let PART.TWO(N) = ANSWER

     case "5"
       write as "Enter new Number Two => ", +
       read NUMBER
       let NUMBER.TWO(N) = NUMBER

     case "D", "d" , "E" , "e" , "Q" , "q" , "X" , "x"
       leave

       default
       endselect
   loop
end       ''COUPLE.MENU


routine EXTRACT.MATRIXES.OF(THIS.COMPONENT) yielding ARRAY1,
                                                     ARRAY2
                                                 and ARRAY3
   define THIS.COMPONENT as an integer variable
   define ARRAY1, ARRAY2, ARRAY3 as 2-dim, real arrays
   reserve ARRAY1 AND ARRAY2 as 3 by 3
   reserve ARRAY3 as 1 by 3

   let ARRAY1(1,1)=1-(L1(THIS.COMPONENT)+L2(THIS.COMPONENT))
   let ARRAY1(2,1) = L2(THIS.COMPONENT)
   let ARRAY1(3,1) = L1(THIS.COMPONENT)
   let ARRAY1(2,2) = 1-L3(THIS.COMPONENT)
   let ARRAY1(3,2) = L3(THIS.COMPONENT)
   let ARRAY1(3,3) = 1

   let ARRAY2(1,1) = 1
   let ARRAY2(1,2) = Mu(THIS.COMPONENT)
   let ARRAY2(2,2) = 1-Mu(THIS.COMPONENT)
   let ARRAY2(3,3) = 1

   let ARRAY3(1,1) = 1
   let ARRAY3(1,2) = 1
 end ''EXTRACT.MATRIXES
```

```
routine INITIALIZE

define I and N as integer variables

 let N = N.COUPLE
 if N.COUPLE gt 0
    for I = 1 to N
      call BUILD.MATRIXES.OF(I)
 else
   call EXTRACT.MATRIXES.OF(1) yielding MATRIX.A(*,*),
                                        MATRIX.B(*,*)
                                   and MATRIX.C(*,*)
 always

  let N = dim.f(MATRIX.A(*,*))


    reserve AT and BT  as N by N
    reserve UT, XT and INPUT.MATRIX as N by 1


for I = 1 to N
  do
    let MATRIX.A(I,I) = MATRIX.A(I,I)-1
    let MATRIX.B(I,I) = MATRIX.B(I,I)-1
  loop


  let XT(1,1) = 1.0
  let T = 0.1


call MATRIXE (MATRIX.A(*,*),T)  yielding E(*,*)
                                          '' Find E matrix

call MATRIXAT (MATRIX.A(*,*),E(*,*)) yielding AT(*,*)
                                        ''Find matrix At

call MATMUL (E(*,*),MATRIX.B(*,*),BT(*,*)) '' Find matrix Bt



open unit 3 for output, file name is "OUT.DAT"
use 3 for output

for I = 1 to dim.f(AT(*,*))
do
 skip 1 output line
 for N = 1 to dim.f(AT(*,*))
  write MATRIX.A(I,N) as D(7,3)
loop
```

```
skip 1 output line
for I = 1 to dim.f(AT(*,*))
do
 skip 1 output line
 for N = 1 to dim.f(AT(*,*))
  write MATRIX.B(I,N) as D(7,3)
loop

skip 1 output line
for I = 1 to dim.f(AT(*,*))
do
 skip 1 output line
 for N = 1 to dim.f(AT(*,*))
  write AT(I,N) as D(10,7)
loop

skip 1 output line
for I = 1 to dim.f(AT(*,*))
do
 skip 1 output line
 for N = 1 to dim.f(AT(*,*))
  write BT(I,N) as D(10,7)
loop

skip 1 output line
for I = 1 to dim.f(AT(*,*))
  write MATRIX.C(1,I) as D(7,3)
close unit 3


release MATRIX.A, MATRIX.B, E, UT


let RELIABILITY = 1.0
let time.v = 0.0              ''Reset simulation clock.

'' Modify the trace icon to match parameters entered by user

  show RELIABILITY with "TRACE.GRF"

  define AXARRAY as 1-dim real array
  let AXARRAY(*) = dary.a(f.display.s(a.reliability))
  let AXARRAY(1) = 0                        '' min value
  let AXARRAY(2) = SIM.TIME                 '' max value
  let AXARRAY(3) = trunc.f(SIM.TIME/20)   '' thick marks
  if SIM.TIME le 10
        let AXARRAY(3) = .5
  endif
    if AXARRAY(5) ne 0
        let AXARRAY(5) = trunc.f(SIM.TIME)   ''grid interval
    endif
    let AXARRAY(6) = trunc.f(SIM.TIME/5)    ''number interval
```

```
        if SIM.TIME gt 99
            let AXARRAY(18) = 4
            let AXARRAY(19) = 0
        endif

      define DEVPTR as a pointer variable
      call devinit.r("VT, GRAPHIC") yielding DEVPTR
      open 7 for input, device = DEVPTR
      open 8 for output, device = DEVPTR
      use 8 for graphic output

        let ERROR.FLAG = 0
        let RELIABILITY = 1.0
        activate a MAINTENANCE now
        activate a SYSTEM.OPERATION now
        activate an OUTPUT now
        activate a RECORDER now


end   ''INITIALIZE
```

```
routine INPUT.MENU

   define CHOICE as an alpha variable
   define ANSWER as an integer variable
   define ANSWER2 as a real variable

   call vbcolor.r(1)
   call vfcolor.r(15)
   let lines.v = 0

   while 0 = 0
    do
      call vclears.r
 print 10 lines with MAINT.PERIOD, REC.PERIOD, SIM.TIME thus


                          MAIN INPUT MENU
                         -----------------

          Current Maintenance Period in Seconds is  = ***.**
          Current Recording Period in Seconds is    = ***.**
          Current Simulation Time in Seconds is     = ***.**


       print 10 lines thus
                          1) Edit a Component
                          2) Edit a Couple
                          3) Change the Maintenance Period
                          4) Change the Recording Period
                          5) Change the Simulation Time

                          R) Run the simulation
                          E) E    the simulation



   call vgotoxy.r(21,0)
   write as "Enter your choice => ", +
   call rcr.r
   read CHOICE as A 1
   call vgotoxy.r(21,0)
   call vclearl.r

   select case CHOICE

   case "1"
    write as "Enter The Component Number To Be Edited => ",+

     read ANSWER
     call COMPONENT.MENU(ANSWER)
```

```
 case "2"
   write as "Enter The Couple Number To Be Edited => ", +
   read ANSWER
   call COUPLE.MENU(ANSWER)

 case "3"
   write as "Enter new Maintenance Period=> ", +
   read ANSWER2
   let MAINT.PERIOD = ANSWER2

 case "4"
   write as "Enter new Recording Period=> ", +
   read ANSWER2
   let REC.PERIOD = ANSWER2

 case "5"
   write as "Enter new Simulation Time => ", +
   read ANSWER2
   let SIM.TIME = ANSWER2

 case "R", "r"
   leave

 case "E", "e", "Q", "q", "X", "x"
   stop
 default
 endselect
loop
   call vclears.r
   print 21 lines thus



          ####################################
          #                                  #
          #  PLEASE WAIT FOR INITIALIZATION  #
          #                                  #
          ####################################



                                  At the end of the program:
                                  -------------------------
                                  First press the Enter.
                                  And wait for the report.

call INITIALIZE
end      ''MENU
```

```
routine INTRO.MESSAGE
   define CHOICE as an alpha variable
   define ANSWER as an integer variable

   call vbcolor.r(1)
   call vfcolor.r(15)
   let lines.v = 0
   call vclears.r
   print 15 lines thus


              A MODEL FOR PREVENTIVE MAINTENANCE
               Written by 1st LT M. E. GUNES

  This program calculates and graphs the resulting reliability of
multicomponent systems under preventive maintenance. The program
solves a linear state space model by discrete time steps. And uses
the transition rates, maintenance period, total simulation time etc.
as inputs which can be entered either by a data file or by keyboard.
  Now you may continue by answering the first question by YES(Y or y)
which means that a data file(under this directory and called
INPUT.DAT) will be used for input. Or you may enter the whole
information by answering the first question by NO(N or n).

   call vgotoxy.r(21,0)
   write as "Are you using a data file (y/n) => ", +
   call rcr.r
   read CHOICE as A 1
   call vgotoxy.r(21,0)
   call vclearl.r

   select case CHOICE

    case "Y" , "y"
      call vclears.r
      call READ.DATA.FILE

    case "N", "n"
    write as "How many components does the system has ? => ",+
        read ANSWER
        let N.COMPONENT = ANSWER
        create every COMPONENT
      call vgotoxy.r(21,0)
      call vclearl.r
      write as "How many couples does the system has ? => ", +
         read ANSWER
         let N.COUPLE = ANSWER
         create every COUPLE
      call INPUT.MENU
    default
   endselect
   end      ''INTRO.MESSAGE
```

```
process MAINTENANCE

   define XXT, C1 and C2 as 2-dim, real arrays
   reserve C1 and C2 as 1 by 1
   let C1(1,1) = 1
   let N = dim.f(INPUT.MATRIX(*,*))
   reserve XXT as N by 1

   while time.v < SIM.TIME
     do
       wait MAINT.PERIOD units

       if ERROR.FLAG = 1
         leave
       always

       call MATMUL(BT(*,*),XT(*,*),INPUT.MATRIX(*,*))
       for I = 1 to N
        let XXT(I,1) = XT(I,1)+INPUT.MATRIX(I,1)
       call MATMUL(MATRIX.C(*,*),XXT(*,*),C2(*,*))

       if C2(1,1) gt C1(1,1) or C2(1,1) gt 1

 let ERROR.MESSAGE = "SORRY! Data Error I Can't Go Further."
         let ERROR.FLAG = 1
         let RELIABILITY = 0.0
         leave
       always

        let  C1(1,1) = C2(1,1)
   loop
 return
end  ''MAINTENANCE


routine MATMUL given A,B and C

  define A,B and C as 2-dim real arrays
  define N,M,S,I,J,K as integer variables
  let N = dim.f(A(*,*))
  let M = dim.f(A(1,*))
  let S = dim.f(B(1,*))

  for I = 1 to N
     for K = 1 to S
   do
     let C(I,K) = 0
         for J = 1 to M
         add A(I,J)*B(J,K) to C(I,K)
   loop
   return
end  ''MATMUL
```

```
routine MATRIXAT given A and E yielding AT

  define I and N as integer variables
  define A, E and AT as 2-dim real arrays


  let N = dim.f(A(*,*))
  reserve AT as N by N


  call MATMUL(A(*,*),E(*,*),AT(*,*))

    for I = 1 to N
      add 1 to AT(I,I)

  return

end  ''MATRIXAT




routine MATRIXE given A and T  yielding E

  define A, AA, AAA and E as 2-dim real arrays
  define T as a real variable
  define I, K, N as integer variables

  let N = dim.f(A(*,*))
  reserve AA,AAA,E as N by N

  call MATMUL(A(*,*),A(*,*),AA(*,*))
  call MATMUL(A(*,*),AA(*,*),AAA(*,*))


    for I = 1 to N
        for K = 1 to N
        do
          add (A(I,K)/2)*T*T to E(I,K)
          add (AA(I,K)/6)*T*T*T to E(I,K)
          add (AAA(I,K)/24)*T*T*T*T to E(I,K)
          if I equal to K
             add T to E(I,K)
          always
        loop

    release AA and AAA

   return

end  ''MATRIXE
```

```
process OUTPUT
   define  Y as 2-dim real array
   reserve Y as 1 by 1

   while time.v < SIM.TIME
    do
     work T units

      call MATMUL(MATRIX.C(*,*),XT(*,*),Y(*,*))

      if ERROR.FLAG = 1 or Y(1,1) gt 1
        let ERROR.FLAG = 1
        let RELIABILITY = 0.0
let ERROR.MESSAGE = "SORRY ! Data Error I can't Go Further."
        leave
      always

        let RELIABILITY = Y(1,1)
     loop

 release Y
end ''OUTPUT


routine READ.DATA.FILE

   define N as an integer variable
   open unit 3 for input, name is "INPUT.DAT"
   use 3 for input

   read N
   create every COMPONENT(N)
   for each COMPONENT
     do
      read L1(COMPONENT), L2(COMPONENT), L3(COMPONENT),
                                         Mu(COMPONENT)
     loop
   read N
   if N gt 0
      create every COUPLE(N)
      for each COUPLE
       do
         read CONNECTION.TYPE(COUPLE), PART.ONE(COUPLE),
NUMBER.ONE(COUPLE), PART.TWO(COUPLE), NUMBER.TWO(COUPLE)
       loop
    always
    read MAINT.PERIOD
    read REC.PERIOD
    read SIM.TIME
 close unit 3
 call INPUT.MENU
end  ''READ.DATA.FILE
```

```
process RECORDER

   open unit 3 for output, file name is "RECORD.DAT"
   use 3 for output

       write time.v as D(5,2)
       write RELIABILITY as D(12,8)
       start new output line


   while time.v < SIM.TIME
     do

       if REC.PERIOD = 0.0
         leave
       always

       if ERROR.FLAG = 1
         leave
       always

       wait REC.PERIOD units

         write time.v as D(5,2)
         wait 0.0001 units
         write RELIABILITY as D(12,8)
         start new output line

     loop


    if REC.PERIOD = 0.0
     wait SIM.TIME+10 units
    always

  close unit 3
  read as /
  call REPORT


 end  ''RECORDER
```

```
routine REPORT

   define CHOICE as an alpha variable

   erase RELIABILITY
   close unit 8
   close unit 7

   call vbcolor.r(1)
   call vfcolor.r(15)
   let lines.v = 0

   while 0 = 0

  do

   call vclears.r
   print 10 lines with ERROR.MESSAGE, RELIABILITY,
                MAINT.PERIOD, REC.PERIOD and SIM.TIME thus


         **************************************************

        Simulation is Ended With
                                      RELIABILITY = *.********

              Maintenance Period(in Seconds) Was = ***.**
              Recording Period(in seconds) Was   = ***.**
              Simulation Time(in Seconds) Was    = ***.**



      print 8 lines thus




       Now you may,
             R) Rerun the same system with new parameters or
             E) Exit the simulation


   let ERROR.MESSAGE = ""
   call vgotoxy.r(19,0)
   write as "Enter your choice => ", +
   call rcr.r
   read CHOICE as A 1
   call vgotoxy.r(21,0)
   call vclearl.r
```

```
  select case CHOICE

   case "R", "r"
     release MATRIX.C, AT, BT, XT and INPUT.MATRIX
     call INPUT.MENU
     leave

   case "E", "e", "Q", "q", "X", "x"
     stop
     default
   endselect
  loop

end  ''REPORT



process SYSTEM.OPERATION

   define DUMMY as 2-dim real array
   define I and N as integer variables

   let N = dim.f(XT(*,*))

   reserve DUMMY as N by 1
   while time.v < SIM.TIME
     do
      work T units

       if ERROR..FLAG = 1
        leave
       always
       call MATMUL(AT(*,*),XT(*,*),DUMMY(*,*))
       for I = 1 to N
        do
         let XT(I,1) = DUMMY(I,1)+INPUT.MATRIX(I,1)
         let INPUT.MATRIX(I,1) = 0.0
        loop

     loop

   release DUMMY
   return

 end  ''SYSTEM.OPERATION
```

Bibliography

1. Ang, Alfredo H-S. and Wilson H. Tang. Probability Concepts in Engineering Planning and Design; Volume II Decision, Risk, and Reliability. New York: John Wiley and Sons, 1984.

2. Clark, A. B. and R. L. Disney. Probability and Random Processes for Engineers and Scientists. New York: John Wiley and Sons, 1970.

3. Fleming, R. M., J. V. Josselyn, L. J. Dolny, and R. L. DeHoff. "Complex System RMA and T, using Markov Models," 1985 Proceedings Annual Reliability and Maintainability Symposium,: 125-130, IEEE 1985.

4. Johnson, S. C., and R. W. Butler. "Automated Generation of Reliability Models," 1988 Proceedings Annual Reliability and Maintainability Symposium,: 17-22, IEEE 1988.

5. Josselyn, J. V., R. E. Fleming, J. A. Frenster, and R. L. DeHoff. "Application of Markov Models for RMA Assessment," 1986 Proceedings Annual Reliability and Maintainability Symposium,: 427-432, IEEE 1986.

6. Kitchin, J. F. "Practical Markov Modeling for Reliability Analysis," 1988 Proceedings Annual Reliability and Maintainability Symposium,: 290-296, IEEE 1988.

7. Reid, J. G. Linear System Fundamentals, Continuous and Discrete, Classic and Modern. McGraw-Hill, Inc. 1983.

8. Riley, J., and B. Putney. "The Risk Menagement Query System," 1986 Proceedings Annual Reliability and Maintainability Symposium,: 358-360, IEEE 1986.

9. Sandler, G. H. System Reliability Engineering, Prentice-Hall, Inc. 1965.

10. Smith, A. M., R. V. Vasudevan, T. D. Matteson, and J. P. Gaertner. "Enhancing Plant Preventive Maintenance via RCM," 1986 Proceedings Annual Reliability and Maintainability Symposium,: 120-121, IEEE 1986.

11. Wild, A. "R&M Design - Problem Definition," 1982 Proceedings Annual Reliability and Maintainability Symposium,: 309-312, IEEE 1986.

VITA

Lieutenant M. Erdogan Gunes [redacted] He graduated from Technical University of Istanbul in July 1982 with the degree of Engineer in Aeronautical Engineering. Upon graduation he was assigned to the Eskisehir Supply and Maintenance Centre of TUAF as second lieutenant. He completed the American Language Course at the same base. He attended a course on Jet Engine Test Cells in Chanute AFB and Seymour Johnson AFB, USA in December 1985. He served as the Jet Engine Maintenance Engineer in Eskisehir Supply and Maintenance Centre of TUAF until entering the School of Engineering, Air Force Institute of Technology, in April 1987.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/AA/88D-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering AF Institute of Technology 6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB Ohio, 45433-6583	7b. ADDRESS (City, State, and ZIP Code) Approved for public release IAW AFR 190-1 12 Jan 1989
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO
11. TITLE (Include Security Classification) MODELLING THE SCHEDULED PREVENTIVE MAINTENANCE AS A LINEAR SYSTEM	
12. PERSONAL AUTHOR(S) GUNES, MEHMET ERDOGAN	
13a. TYPE OF REPORT MS THESIS	13b. TIME COVERED FROM _____ TO _____
14. DATE OF REPORT (Year, Month, Day) 881123	15. PAGE COUNT 86
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	03		Preventive Maintenance
01	03		Markov Processes
			Linear Systems

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: David G. Robinson, Ph.D., Major, USAF
Assistant Professor, Dep. of Aeronautic and Astronautics

The purpose of this research is to develop a model of scheduled preventive maintenance that will allow the design engineer and logistics planners to predict future, long term system reliability based on scheduled preventive maintenance with different maintenance periods and capabilities. The model uses Markov processes but formulates the equations as a linear, state variable control system. This technique will simplify the Markov models of large and complex systems. And will lead to a computer model which runs fast, has low operating cost and consequently provides higher sensitivity analyses.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL DAVID G. ROBINSON, Major, USAF	22b. TELEPHONE (Include Area Code) 513-785-6998 / 22c. OFFICE SYMBOL AFIT/ENY

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED